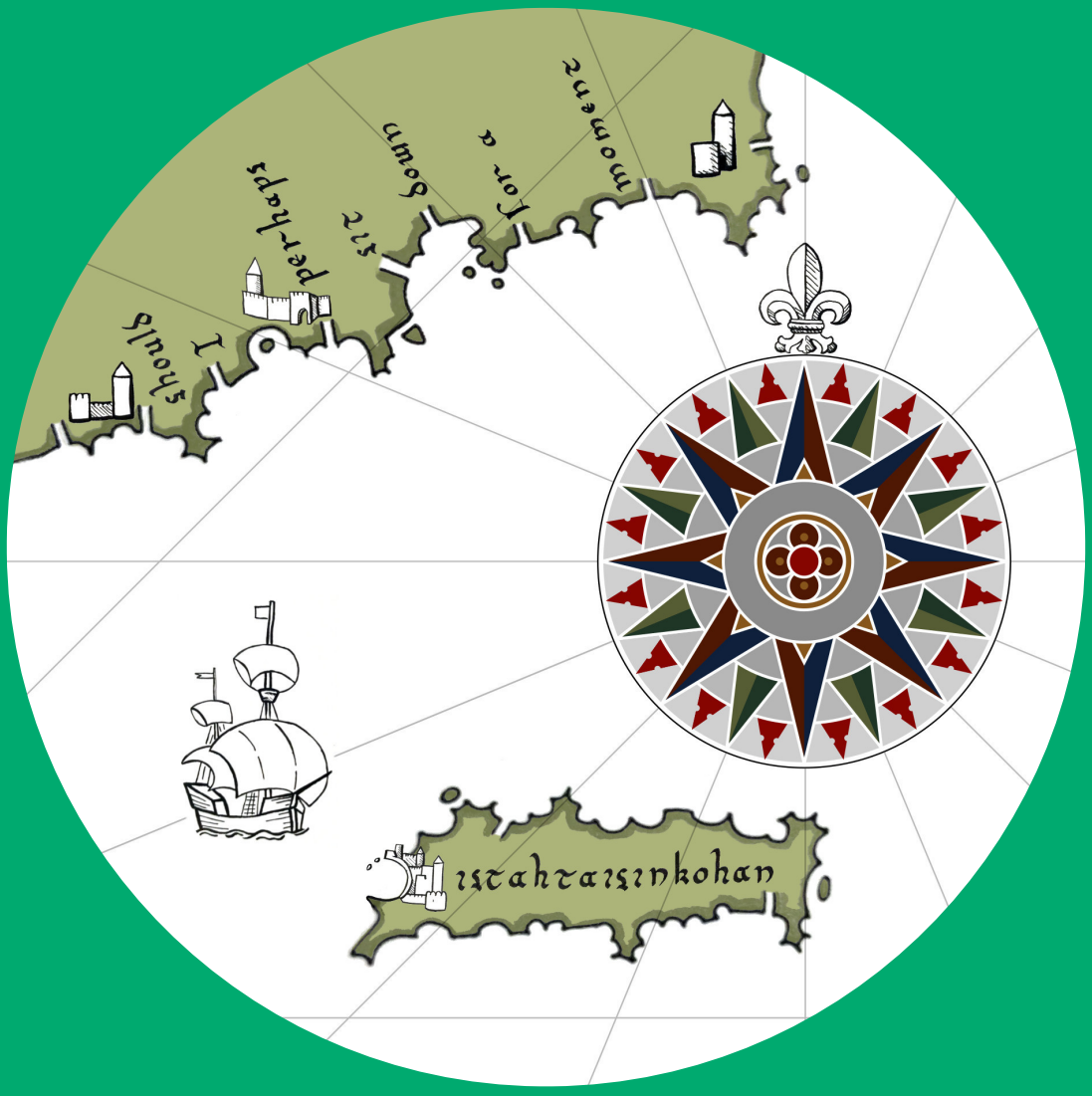
Department of Signal Processing and Acoustics

Machine translation into morphologically rich low-resource languages

Stig-Arne Grönroos



Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 202/2020

Machine translation into morphologically rich low-resource languages

Stig-Arne Grönroos

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Electrical Engineering, Remote connection link (Zoom), on 18 January 2021 at 12:00.

Aalto University
School of Electrical Engineering
Department of Signal Processing and Acoustics
Speech and Language Processing research group

Supervising professor
Prof. Mikko Kurimo, Aalto University, Finland

Thesis advisor
Dr. Sami Virpioja, University of Helsinki, Finland

Preliminary examiners
Dr. Maja Popovic, ADAPT Centre at Dublin City University, Ireland
Prof. Robert Östling, Stockholms universitet, Sweden

Opponents
Prof. Rico Sennrich, University of Zürich, Switzerland

Aalto University publication series
DOCTORAL DISSERTATIONS 202/2020

ISBN 978-952-64-0168-3 (printed)
ISBN 978-952-64-0169-0 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-64-0169-0

Images: Cover image © Stig-Arne Grönroos

Unigrafia Oy
Helsinki 2020

Finland

NORDIC SWAN ECOLABEL
Printed matter
4041-0619

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Stig-Arne Grönroos

Name of the doctoral dissertation
Machine translation into morphologically rich low-resource languages

Publisher School of Electrical Engineering

Unit Department of Signal Processing and Acoustics

Series Aalto University publication series DOCTORAL DISSERTATIONS 202/2020

Field of research Speech and Language Technology

Manuscript submitted 14 August 2020 **Date of the defence** 18 January 2021

Permission for public defence granted (date) 15 October 2020 **Language** English

☐ **Monograph** ☒ **Article dissertation** ☐ **Essay dissertation**

Abstract

Machine translation is an important natural language processing application, enabling widened access to information, cultural interchange, and business opportunities in a multilingual world. Driven by research into deep neural networks, machine translation has recently made rapid advances, particularly in the fluency of the translation output. As the methods tend to be data-hungry, high-resource languages have benefited more than low-resource ones.

In this work, the aim is to improve machine translation into low-resource morphologically rich languages. Rich morphology leads to a combinatorial explosion in the number of word forms, resulting in very large vocabularies, containing many poorly modeled rare words. This thesis addresses these challenges with multiple approaches. The focus is on methods for segmenting words into subwords, to get more frequent and thus easier learned representations, and to increase the symmetry between languages. It is important to exploit additional resources from related tasks, such as parallel data from related high-resource language pairs and monolingual data from both low- and high-resource languages. Useful auxiliary data sets for multimodal translation can be found from captioning and text-only translation tasks. The methods for exploiting this auxiliary data include cross-lingual learning and data augmentation e.g. using denoising sequence autoencoders and subword regularization. Learning setups used in the thesis include using unsupervised and language-independent methods, using active learning to guide an annotation effort to produce more informative data, and using scheduled multi-task learning to improve cross-lingual transfer.

Contributions of the thesis include five novel segmentation methods: Morfessor FlatCat, Omorfi-restricted Morfessor, Cognate Morfessor, Morfessor EM+Prune, and a semi-supervised neural method. An active learning strategy for Morfessor FlatCat is presented. Evaluation of segmentation quality is performed using both intrinsic and extrinsic automatic methods. Morfessor EM+Prune finds models with both lower cost and better quality in unsupervised segmentation than Morfessor Baseline. Active learning is superior to random selection for collecting annotations. The best performance in semi-supervised segmentation is achieved when using Morfessor FlatCat segmentations as features in a conditional random field.

Contributions to machine translation include a target-side multi-task learning scheme, and scheduled multi-task learning with a denoising sequence autoencoder. LeBLEU, an evaluation measure suitable for morphologically rich languages is presented. Evaluation of translation quality is performed using both automatic and human evaluation. When resources are scarce, the most important auxiliary data comes from related languages. Other types of auxiliary data, such as monolingual corpora, are also beneficial and the gains are partly cumulative.

Keywords machine translation, morpheme segmentation, subwords, unsupervised learning, semi-supervised learning, transfer learning, multi-task learning, active learning, deep neural networks, low-resource languages, denoising sequence autoencoder

ISBN (printed) 978-952-64-0168-3 **ISBN (pdf)** 978-952-64-0169-0

ISSN (printed) 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Helsinki **Location of printing** Helsinki **Year** 2020

Pages 21 **urn** http://urn.fi/URN:ISBN:978-952-64-0169-0

Aalto-yliopisto

Tiivistelmä

Aalto-yliopisto, PL 11000, 00076 Aalto www.aalto.fi

Tekijä
Stig-Arne Grönroos

Väitöskirjan nimi
Konekäännös morfologisesti rikkaisiin resurssiniukkoihin kieliin

Julkaisija Sähkötekniikan korkeakoulu

Yksikkö Signaalinkäsittelyn ja akustiikan laitos

Sarja Aalto University publication series DOCTORAL DISSERTATIONS 202/2020

Tutkimusala Puhe- ja kieliteknologia

Käsikirjoituksen pvm 14.08.2020 **Väitöspäivä** 18.01.2021

Väittelyluvan myöntämispäivä 15.10.2020 **Kieli** Englanti

☐ **Monografia** ☒ **Artikkeliväitöskirja** ☐ **Esseeväitöskirja**

Tiivistelmä

Konekäännös on tärkeä luonnollisten kielten käsittelyn sovellus, joka mahdollistaa entistä laajemman pääsyn tietoon monikielisessä maailmassa, sekä edesauttaa kulttuurista vuorovaikutusta ja liiketoimintaa. Konekäännös on kehittynyt nopeasti viimeaikoina syviin neuroverkkoihin kohdistuvan tutkimuksen ansiosta. Etenkin käännöksen kielellinen sujuvuus on edistynyt. Koska menetelmät edellyttävät suuria datamääriä, kehitys on keskittynyt hyvin resursoiduille kielille.

Tämän väitöskirjan tavoitteena on edistää konekäännöstä kun kohdekielenä on morfologisesti rikas kieli, jolle on saatavilla niukasti resursseja. Kielen rikas morfologia johtaa sananmuotojen määrän kombinatoriseen räjähdykseen tuottaen erittäin suuria sanastoja. Harvinaisia sananmuotoja on vaikea mallintaa. Väitöskirjassa näihin haasteisiin vastataan hyödyntäen useaa lähestymistapaa. Pääasiallinen lähestymistapa on sanojen pilkonta osiin. Pilkonnan avulla saadut esitystavat ovat helpompia mallintaa, minkä lisäksi pilkonnalla voidaan parantaa kielten välistä symmetriaa. On tärkeää hyödyntää resursseja läheisistä sovelluksista, esimerkiksi sukulaiskielten rinnakkaista tekstiä sisältävistä aineistoista, sekä yksikielisistä aineistoista. Väitöskirjassa käytetään kieliriippumattomia menetelmiä ja erilaisia koneoppimisasetelmia, kuten ohjaamatonta oppimista. Apuaineistoja hyödynnetään käyttämällä monikielistä oppimista, datan rikastamista, kohinaa poistavaa sekvenssiautoenkooderia, sekä pilkontaregularisointia. Aktiivista koneoppimista käytetään tehokkaampaan annotaatioiden keräämiseen, ja aikataulutettua monen tehtävän oppimista monikielisen oppimisen tehostamiseen.

Väitöskirjassa esitellään viisi uutta menetelmää sanojen pilkontaan: Morfessor FlatCat, Omorfi-restricted Morfessor, Cognate Morfessor, Morfessor EM+Prune, sekä puoliohjattu neuroverkkoihin perustuva menetelmä. Morfessor FlatCat -menetelmälle esitellään aktiivisen koneoppimisen strategia. Pilkonnan laatua arvioidaan sekä suorilla että epäsuorilla automaattisilla evaluaatioilla. Morfessor EM+Prunen löytämillä malleilla on sekä alempi kustannusfunktion arvo että parempi pilkonnan laatu kuin Morfessor Baseline -menetelmällä. Aktiivinen koneoppiminen on satunnaista valintaa parempi annotaatioiden keräämiseen. Puoliohjatussa pilkonnassa paras laatu saavutetaan käyttämällä Morfessor FlatCatin pilkontoja piirteinä ehdollisessa satunnaiskentässä.

Konekäännöksen menetelminä esitellään kohdekielen puolella tapahtuva monen tehtävän oppiminen sekä aikataulutettu monen tehtävän oppiminen, joka hyödyntää kohinaa poistavaa sekvenssiautoenkooderia. Lisäksi esitellään morfologisesti rikkaille kohdekielille soveltuva evaluaatiomenetelmä, LeBLEU. Konekäännöksen laatua arvioidaan sekä automaattisilla että ihmisarvioihin perustuvilla menetelmillä. Kun tavoitteena olevan käännöstehtävän resurssit ovat vähäisiä, tärkeimmät apuaineistot ovat sukulaiskielet. Myös yksikielisistä aineistoista on hyötyä.

Avainsanat konekäännös, morfeeemipilkonta, ohjaamaton oppiminen, puoliohjattu oppiminen, siirto-oppiminen, monen tehtävän oppiminen, aktiivinen koneoppiminen, syvät neuroverkot, resurssiniukat kielet, kohinaapoistava sekvenssiautoenkooderi

ISBN (painettu) 978-952-64-0168-3 **ISBN (pdf)** 978-952-64-0169-0

ISSN (painettu) 1799-4934 **ISSN (pdf)** 1799-4942

Julkaisupaikka Helsinki **Painopaikka** Helsinki **Vuosi** 2020

Sivumäärä 21 **urn** http://urn.fi/URN:ISBN:978-952-64-0169-0

Aalto-universitetet

Sammandrag

Aalto-universitetet, PB 11000, 00076 Aalto www.aalto.fi

Författare
Stig-Arne Grönroos

Doktorsavhandlingens titel
Maskinöversättning till morfologiskt rika resursfattiga språk

Utgivare Högskolan för elektroteknik

Enhet Institutionen för signalbehandling och akustik

Seriens namn Aalto University publication series DOCTORAL DISSERTATIONS 202/2020

Forskningsområde Tal- och språkteknologi

Inlämningsdatum för manuskript 14.08.2020 **Datum för disputation** 18.01.2021

Beviljande av disputationstillstånd (datum) 15.10.2020 **Språk** Engelska

☐ **Monografi** ☒ **Artikelavhandling** ☐ **Essäavhandling**

Sammandrag

Maskinöversättning är en viktig uppgift i behandling av naturliga språk. Maskinöversättning möjliggör ökad tillgång till information, kulturellt utbyte och affärsmöjligheter i en flerspråkig värld. Forskning i djupa neuralnät har lett till en snabb förbättring i maskinöversättningens kvalitet, med speciell betoning på språkligt flyt. På grund av att metoderna i regel förutsätter stora datamängder, har förbättringen gynnat resursrika språk mest.

Syftet med denna avhandling är att förbättra maskinöversättning till resursfattiga morfologiskt rika språk. Rik morfologi leder till en kombinatorisk explosion i antalet ordformer, vilket resulterar i stora vokabulärer med flera sällsynta ord som är svåra att modellera. I avhandlingen utnyttjas flera strategier för att ta itu med dessa utmaningar. Den viktigaste är segmentering av ord till mindre beståndsdelar, för att få oftare förekommande representationer som är lättare att modellera, och för att öka symmetrin mellan språken. Det är viktigt att utnyttja resurser från relaterade uppgifter, till exempel parallella data för besläktade resursrika språkpar, och enspråkiga data för både resursrika och resursfattiga språk. I multimodal översättning kan sekundära data hittas från generering av bildtexter och text-baserad översättning. Metoder för att utnyttja dessa sekundära data innefattar tvärspråklig inlärning, augmentering av data, användande av brusdämpande sekvensautoenkoder, och segmenteringsbaserad regularisering. Uppställningar för maskininlärning som används i avhandlingen innefattar ostyrd maskininlärning och språkoberoende metoder, användande av aktiv maskininlärning för att förbättra annotering, samt schemalagd inlärning med flera uppgifter för att förbättra tvärspråklig inlärning.

Avhandlingen presenterar fem nya segmenteringsmetoder: Morfessor FlatCat, Omorfi-restricted Morfessor, Cognate Morfessor, Morfessor EM+Prune, och en neural metod för halvstyrd inlärning. En strategi för aktiv maskininlärning för Morfessor FlatCat presenteras. För utvärdering av segmenteringskvalitet används både direkt och indirekt automatisk evaluation. Morfessor EM+Prune hittar modeller med lägre kostnad och bättre prestanda än Morfessor Baseline. Aktiv inlärning är bättre än slumpmässigt urval för att samla annoteringar. För halvstyrd inlärning är det bäst att mata in segmenteringar från Morfessor FlatCat som särdrag i ett betingat slumpfält.

Som metoder för maskinöversättning presenteras en typ av inlärning med flera uppgifter på målspråkets sida, och schemalagd inlärning med flera uppgifter som använder brusdämpande sekvensautoenkoder. För utvärdering av maskinöversättning till morfologiskt rika språk presenteras LeBLEU. Kvaliteten av maskinöversättningen utvärderas både automatiskt och av människor. När det finns begränsat av resurser, är besläktade språk den viktigaste formen av sekundära data. Användning av enspråkiga sekundära data ger en delvis kumulativ nytta.

Nyckelord maskinöversättning, morfologisk segmentering, ostyrd inlärning, halvstyrd inlärning, överförd inlärning, inlärning med flera uppgifter, aktiv maskininlärning, djupa neuralnät, resursfattiga språk, brusdämpande sekvensautoenkoder

ISBN (tryckt) 978-952-64-0168-3 **ISBN (pdf)** 978-952-64-0169-0

ISSN (tryckt) 1799-4934 **ISSN (pdf)** 1799-4942

Utgivningsort Helsingfors **Tryckort** Helsingfors **År** 2020

Sidantal 21 **urn** http://urn.fi/URN:ISBN:978-952-64-0169-0

Preface

> “Y. That perfect letter. The wishbone, fork in the road, empty wineglass. The question we ask over and over. (Marjorie Celona, 2012) ”

The work for this thesis was done in the Department of Signal Processing and Acoustics at Aalto University. I received funding from the Aalto ELEC Doctoral School, the Academy of Finland in the COIN Centre of Excellence, the European Union's Horizon 2020 Research and Innovation Programme in the Simple4All and MeMAD projects, and a personal grant from Aalto-yliopiston tekniikan tukisäätiö for finalizing the dissertation. The many computationally intensive experiments in the thesis were made possible by the resources of the Aalto Science-IT project.

Institutions grant opportunities, but it is the people who truly enable growth and collaboration. First, I want to thank my supervisor Prof. Mikko Kurimo for his efforts to make everything run smoothly and efficiently, and a real skill for connecting the right people and ideas. I am thankful for the encouragement, advice, and believing in me even when I came with ideas outside his primary area of expertise. Also, never needing to worry about the money running out allowed me to fully focus on the research.

A big thank you to my instructor Dr. Sami Virpioja, whose work was a major influence on my selection of research topics. Thank you for patiently listening to my wild ideas, providing invaluable help sorting out the wheat from the chaff, and all the guidance in tying the whole package together, when my curiosity had taken my work in too many directions. Sami taught me a great deal about good research practices and how to present the ideas and results in writing, which, in short, made me a better scientist.

I want to thank the preliminary examiners Dr. Maja Popović and Dr. Robert Östling for all the thoughtful comments and valuable suggestions on how to improve the thesis. Apart from my supervisor, instructor, and pre-examiners, other people who have read parts of the manuscript and given valuable feedback are Dr. Mittul Singh, and Ekaterina Voskoboinik.

My sincerest gratitude to all my co-authors (in alphabetical order) for the opportunity to work together: Katri Hiovain, Benoit Huet, Kristiina Jokinen, Oskar Kohonen, Mikko Kurimo, Jorma Laaksonen, Bernard Merialdo, Phu Pham, Ilona Rauhala, Teemu Ruokolainen, Kairit Sirts, Mats Sjöberg, Peter Smit, Umut Sulubacak, Jörg Tiedemann, Raphaël Troncy, Sami Virpioja, and Raúl Vázquez. It has certainly been a privilege to work with such great people.

I am forever grateful to the Finnish educational system. I am happy to pay

my taxes to support it, as I think everyone should, including multinationals and wealthy individuals.

All the anonymous reviewers and editors who have invested their time in improving my publications with their suggestions deserve to be thanked. Everyone—rich or poor—should have access to education and the fruits of science. Scientific publishing should create an openly available ever growing scientific discussion, for the good of humanity, rather than for profit. The rent-seeking behavior of some incumbent academic publishers is not aligned with this goal.

I've had the good fortune of working in a great research group. In particular, I would like to thank Dr. Seppo Enarvi, Dr. Dhananjaya N. Gowda, Dr. Reima Karhila, Dr. Oskar Kohonen, Katri Leino, Dr. Ulpu Remes, Aku Rouhe, Dr. Teemu Ruokolainen, Dr. Mittul Singh, Dr. Peter Smit, Dr. Sami Virpioja, and Katja Voskoboinik. Thank you for the foosball and tabletop games, stimulating lunchtime discussions on widely ranging topics, and all the rigorous experimentation that went into finding the perfect coffee. I consider you friends as much as colleagues. I also want to thank Prof. Jörg Tiedemann and his group, for arranging numerous events that have fostered fruitful discussions, and for many valuable collaborations. On conference trips I have had great conversations that gave me fresh ideas with many people, unfortunately too many to list here.

Finally, I want to thank family, relatives, and friends outside of the university. From my parents, Riitta and Hasse, I inherited the thirst of curiosity that drives my work. Riitta's example planted the seed idea of going into research. My brother Per-Oskar has contributed valuable, although sometimes stinging, advice on graphical aspects[1] of my work, and my performance in general. The biggest thank you of all belongs to the love of my life, my wife Riikka, for all the support and patience when I had times of doubt, and for celebrating with me the times of joy. Enduring my many evenings spent deep in thought, both at the office and at home, she has always succeeded in making sure that work is not the most important thing in my life. This work is dedicated to you.

Espoo, December 19, 2020,

Stig-Arne Grönroos

[1] All unfortunate typographical choices in this thesis are naturally my own.

Contents

List of Publications

> “Two roads diverged in a wood, and I—
> I took the one less traveled by,
> And that has made all the difference.
> (Robert Frost, 1916)”

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, Dublin, Ireland, pages 1177–1185, Aug 2014.

II Teemu Ruokolainen, Oskar Kohonen, Kairit Sirts, Stig-Arne Grönroos, Mikko Kurimo, and Sami Virpioja. A comparative study on minimally supervised morphological segmentation. *Computational Linguistics*, Vol. 42, No. 1, pages 91–120, Mar 2016.

III Stig-Arne Grönroos, Katri Hiovain, Peter Smit, Ilona Rauhala, Kristiina Jokinen, Mikko Kurimo, and Sami Virpioja. Low-resource active learning of morphological segmentation. *Northern European Journal of Language Technology*, Vol. 4, Article 4, pages 47–72, Mar 2016.

IV Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. North Sámi morphological segmentation with low-resource semi-supervised sequence labeling. In *Proceedings of the fifth Workshop on Computational Linguistics for Uralic Languages*, Tartu, Estonia, pages 15–26, Jan 2019.

V Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. Morfessor EM+Prune: Improved subword segmentation with expectation maximization and pruning. In *Proceedings of the 12th Language Resources and Evaluation Conference*, Marseilles, France, 2020, pages 3944–3953, May 2020.

VI Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. Hybrid morphological segmentation for phrase-based machine translation. In *Proceedings of the First Conference on Machine Translation*, Berlin, Germany, pages 289–295, Aug 2016.

VII Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. Tuning phrase-based segmented translation for a morphologically complex target language. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, Lisbon,

Portugal, pages 105–111, Sep 2015.

VIII Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. Cognate-aware morphological segmentation for multilingual neural translation. In *Proceedings of the Third Conference on Machine Translation*, Brussels, Belgium, pages 386–393, Oct 2018.

IX Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. Transfer learning and subword sampling for asymmetric-resource one-to-many neural translation. Accepted for publication in *Machine Translation*, Vol. 34, Oct 2020.

X Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. Extending hybrid word-character neural machine translation with multi-task learning of morphological analysis. In *Proceedings of the Second Conference on Machine Translation*, Copenhagen, Denmark, pages 296–302, Sep 2017.

XI Stig-Arne Grönroos, Benoit Huet, Mikko Kurimo, Jorma Laaksonen, Bernard Merialdo, Phu Pham, Mats Sjöberg, Umut Sulubacak, Jörg Tiedemann, Raphael Troncy, and Raúl Vázquez. The MeMAD submission to the WMT18 multimodal translation task. In *Proceedings of the Third Conference on Machine Translation*, Brussels, Belgium, pages 603–611, Oct 2018.

XII Sami Virpioja and Stig-Arne Grönroos. LeBLEU: N-gram-based translation evaluation score for morphologically complex languages. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal, pages 411–416, Sep 2015.

Author's Contribution

> “ I'm being quoted to introduce something, but I have no idea what it is and certainly don't endorse it.
> (Randall Munroe, 2018) ”

Publication I: “Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology”

The present author designed and implemented the algorithm described in the article, and conducted the experiments. The present author was also the main writer of the article.

Publication II: “A comparative study on minimally supervised morphological segmentation”

The experiments using Morfessor FlatCat were conducted and analyzed by the present author. The present author also contributed to writing the article.

Publication III: “Low-resource active learning of morphological segmentation”

The present author conducted the experiments, and contributed to the analysis of the results and overall coordination of the work. The present author was also the main writer of the article.

Publication IV: “North Sámi morphological segmentation with low-resource semi-supervised sequence labeling”

The present author designed and implemented the algorithm described in the article, and conducted the experiments. The present author was also the main writer of the article.

Publication V: "Morfessor EM+Prune: Improved subword segmentation with expectation maximization and pruning"

The present author designed and implemented the algorithm described in the article, and conducted the experiments. The present author was also the main writer of the article.

Publication VI: "Hybrid morphological segmentation for phrase-based machine translation"

The present author designed and implemented the algorithm described in the article, and conducted the experiments. The present author was also the main writer of the article.

Publication VII: "Tuning phrase-based segmented translation for a morphologically complex target language"

The present author designed and implemented the algorithm described in the article, and conducted the experiments. The present author was also the main writer of the article.

Publication VIII: "Cognate-aware morphological segmentation for multilingual neural translation"

The present author designed and implemented the algorithm described in the article, and conducted the experiments. The present author was also the main writer of the article.

Publication IX: "Transfer learning and subword sampling for asymmetric-resource one-to-many neural translation"

The present author designed and implemented the algorithm described in the article, and conducted the experiments. The present author was also the main writer of the article.

Publication X: "Extending hybrid word-character neural machine translation with multi-task learning of morphological analysis"

The present author designed and implemented the algorithm described in the article, and conducted the experiments. The present author was also the main writer of the article.

Publication XI: "The MeMAD submission to the WMT18 multimodal translation task"

The present author designed and implemented the primary system, and conducted a majority of the experiments. The present author was also the main writer of the article.

Publication XII: "LeBLEU: N-gram-based translation evaluation score for morphologically complex languages"

The present author implemented the algorithm described in the article, and conducted the experiments. The present author also participated in designing the algorithm, and in writing of the article.

List of abbreviations

> “It is often easier to discover a truth than to assign to it its proper place (de Saussure, 1916) ”

AI	Artificial Intelligence
AL	Active Learning
ANN	Artificial Neural Network
BLEU	BiLingual Evaluation Understudy
BMES	Beginning–Middle–End–Single
BPE	Byte Pair Encoding
BPR	Boundary Precision and Recall
BT	Back-Translation
CRF	Conditional Random Field
DSAE	Denoising Sequence AutoEncoder
EBMT	Example Based Machine Translation
EM	Expectation Maximization
FAHQT	Fully Automatic High-Quality Translation
FF	Feed-Forward network
FST	Finite State Transducer
GOFAI	Good Old-Fashioned Artificial Intelligence
GPU	Graphical Processing Unit
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
HRL	High-Resource Language
IA	Item-and-Arrangement
IP	Item-and-Process
LM	Language Model
LR-MNMT	Low-Resource Multilingual NMT
LRL	Low-Resource Language
LSTM	Long Short-Term Memory
LSV	Letter Successor Variety

MAP	Maximum A Posteriori
MBR	Minimum Bayes Risk
MDL	Minimum Description Length
MEMM	Maximum Entropy Markov Model
ML	Machine Learning
MNMT	Multilingual Neural Machine Translation
MRL	Morphologically Rich Language
MT	Machine Translation
MTL	Multi-Task Learning
NLP	Natural Language Processing
NMT	Neural Machine Translation
NNLM	Neural Network Language Model
OOV	Out-Of-Vocabulary
ORM	Omorfi-Restricted-Morfessor
PB-SMT	Phrase-Based Statistical Machine Translation
RNN	Recurrent Neural Network
ReLU	Rectified Linear Unit
Seq2seq	Sequence-to-sequence
SGD	Stochastic Gradient Descent
SMT	Statistical Machine Translation
WP	Word-and-Paradigm

CZE	Czech
DAN	Danish
ENG	English
EST	Estonian
FIN	Finnish
NOB	Norwegian Bokmål
SLO	Slovak
SME	North Sámi
SWE	Swedish

PRE	Prefix
STM	Stem
SUF	Suffix
NON	Non-morpheme

List of symbols and notations

> “Behind a superficial appearance of simplicity, there is concealed a perfect hornet’s nest of bizarre and arbitrary usages (Edward Sapir, 1949)”

$A, B, \ldots$	Random variables, integers
$a, b, \ldots,$	Scalars
$\boldsymbol{a}, \boldsymbol{b}, \ldots$	Vectors, sequences
$\boldsymbol{A}, \boldsymbol{B}, \ldots$	Matrices, tensors
$\mathrm{A}, \mathrm{B}, \ldots$	Sets
$\boldsymbol{a}$	Alignment between sequences $\boldsymbol{s}$ and $\boldsymbol{t}$
$\mathscr{A}$	Training pool for active learning
$\arg\min_x f(x)$	Argument x minimizing the function f
$\arg\max_x f(x)$	–”– maximizing –”–
$\mathrm{C}(A)$	The occurrence count of event A
$\boldsymbol{D}$	Data
$\boldsymbol{D}^{(L)}$	Labeled data
$\boldsymbol{D}^{(T)}$	Data for task $\boldsymbol{T}$
$\boldsymbol{D}^{(U)}$	Unlabeled data
$\mathfrak{e}$	String edit operation
$\boldsymbol{h}$	Hidden state
I	Indicator function
$\boldsymbol{K}$	Key matrix
L	Lexicon, vocabulary
$\mathrm{L}(\boldsymbol{\theta}, \boldsymbol{D})$	Loss (cost) function
$\mathbf{M}$	Model
m	Morph, subword
$\mathbb{N}$	The natural numbers
$P(A)$	Probability of event A
$P(x)$	Probability distribution for X
$\boldsymbol{q}$	Query vector
$\mathbb{R}$	The real numbers

$\boldsymbol{s} = s_{1:I} = (s_1, \ldots, s_I) \in \Sigma_{\text{src}}^I$	Source sequence
$\boldsymbol{t} = t_{1:J} = (t_1, \ldots, t_J) \in \Sigma_{\text{trg}}^J$	Target sequence
$V = \lvert \mathrm{L} \rvert$	Vocabulary size
$\boldsymbol{V}$	Value matrix
$\boldsymbol{W}$	Weight matrix
$\boldsymbol{X}$	Features
$\boldsymbol{Y}$	Labels
$\mathscr{Y}$	Hypothesis space, label space
$\alpha, \beta, \ldots$	Scalars
δ	The Dirac function
ϵ	Empty (end-of-word) character
ϵ	Learning rate
$\boldsymbol{\theta}$	Model parameters
Θ	Parameter space
Σ^*	All sequences of an alphabet
ϕ^{-1}	Detokenization function
Ψ	The digamma function
$[\boldsymbol{a}; \boldsymbol{b}]$	Concatenation of sequences $\boldsymbol{a}$ and $\boldsymbol{b}$
x^*	Optimal value
$\hat{x}$	Estimated value
$\tilde{x} = \mathrm{C}(x)$	Value corrupted by noise
$\lvert \boldsymbol{x} \rvert$	Length of sequence $\boldsymbol{x}$
$\lvert x \rvert$	Absolute value of scalar x
$\lceil x \rceil$	Ceiling rounding function
∇f	Gradient of f
$\otimes$	Elementwise multiplication
$\mathbf{1}(x)$	The identity function
$\mapsto$	Mapping
$+$	Morph boundary
$\_$	Word boundary
$\langle \text{UNK} \rangle$	Special *unknown* token

1. Introduction

> “Language is the meta-language
> (Eldon G. Lytle, 1974) ”

One of the distinguishing characteristics of humans is the propensity to use language for communication. We humans have put this skill to great use, and in the process we have created ca 7000 (Simons and Fennig, 2018) naturally evolved languages that humans use in communication with each other, called ***natural languages***. Some examples of natural languages are English, Finnish, and Latin. Natural languages can be contrasted against constructed, controlled, and formal languages. ***Constructed*** languages, e.g. Lojban, Esperanto, and Sindarin, were designed for some particular use. One prominent subset of constructed languages are the proposed international auxiliary languages, intended to be learned by everyone to facilitate global communication. ***Controlled*** languages are structured sub-languages that add constraints limiting the expressiveness and ambiguity of a natural language. ***Formal*** languages, including programming languages such as Python and C, are much less ambiguous and more structured, making them very suitable for automatic processing.

Human language use is viewed as either a sign of intelligence, or even a central component in creating intelligence. Being able to build machines with language ability is a major milestone in the quest for artificial intelligence. Turing (1950) proposed a famous test for evaluating the ability of a machine to exhibit intelligent behavior. In the test, a human evaluator must determine which of two test subjects is a human and which is a machine, based only on textual communication. The ELIZA chatbot (Weizenbaum, 1966) cast some doubt on the effectiveness of humans as judges in a Turing test. ELIZA was designed to lead the conversation in directions where the human did most of the work, and it could fool people into overestimating its ability. On a positive note, ELIZA showed that even very shallow language ability can be useful under some conditions.

To solve general natural language tasks is a very challenging problem, which requires both understanding the meaning of ambiguous statements, and the ability to generate replies in the form of natural language. Humans solve these tasks nearly effortlessly, and it is easy to underestimate the difficulty of finding artificial solutions to them.

Machine translation is an example of a natural language task that requires both understanding of source text and generation of target text. The dream of building a universal translation device, like the Babel fish from Douglas Adams'

The Hitchhiker's Guide to the Galaxy, has captivated the minds of many thinkers throughout history. Such a device would translate any speech and text into the native language of the user. The translation would occur in real-time as the user encounters material in any foreign language.

Effective general translation would be beneficial to society in many ways. Translation could enable increased access to the wealth of digital information collected on-line, regardless of which languages the reader understands. As most of this information is in English, the greatest benefits would fall to non-English speakers. Translation can open new business opportunities in many large multilingual markets, including the EU. In addition to how machine translation can help humans communicate with each other, methods developed in machine translation can power advances in natural language understanding, opening new possibilities for human-computer interaction.

Machine translation has made rapid advances in the last decade, both academically and commercially. The improvement is most visible in increased ***fluency***, meaning that the translation output is more often grammatically correct and native sounding. The main reason for this success is new methods applying deep neural networks to machine translation. The rapid digitalization of society has been another trend driving improved performance during the last decades. The availability of parallel training corpora suitable for training machine translation systems has rapidly increased.

Despite the recent success, the task of machine translation is far from solved. Current state-of-the-art methods require very large amounts of data, preferably from the domain of the intended use. The hunger for data is in part due to insufficient language understanding and ability to generalize from that understanding. Translation of a sentence is unlikely to succeed unless the training data contains examples discussing the same topics, and examples using similar expressions.

Languages are not equal with regard to resources. Languages vary in the number of speakers, but also in the wealth of the nations in which the language is used, and the degree of digitalization. When ranked by the number of first-language speakers, the top 8 languages cover 40% of speakers, while 94.3% of languages (all except the top 308) have less than 1 million speakers each (Simons and Fennig, 2018). To mention the language coverage of some example resources at the time of writing, Universal Dependencies are available in 92 languages (Universal Dependencies contributors, 2020), Google Translate in 108 languages, and the New Testament in over 1000 languages. The resource with the largest coverage is the Automatic Similarity Judgment Program (Wichmann et al., 2020), which covered 5499 languages at the time of writing.

The amount of data needed can also vary based on language characteristics. Rich, productive morphology leads to a combinatorial explosion in the number of word forms. Therefore, a larger corpus is required to reach the same coverage of word forms. Data sparsity can also arise within a language, e.g. geographically from dialects, or temporally through changing language use. Domains of text with specific vocabularies and styles of expression also contribute to sparsity.

The differences in resources lead to a scarcity in available language technology to use as parts of applications. Having engineers hand-craft rule-based systems separately for each language requires large amounts of human effort, easily exceeding the available resources considering the large number of languages in the world. Such methods do not require large corpora, only human effort. An alternative is to use ***machine learning***, where the parameters of a statistical model are fit to training data. However, statistical methods work best with large amounts of data, which can be difficult to find or collect for most of the world's languages. Therefore, developing ***language-independent*** methods to improve the effectiveness of learning from small data is very valuable for the large number of low-resource languages. Examples of such methods include segmenting words into ***subwords*** to get more frequent and thus easier learned representations, using ***cross-lingual learning*** to leverage training data from related high-resource languages, exploiting monolingual and other auxiliary data via ***data augmentation***, and using ***active learning*** to guide an annotation effort to produce maximally informative data.

From an ethical perspective, translation into low-resource languages comes with unique concerns. When making public machine translation into minority languages, high quality must be demanded, even though low resources make it difficult to achieve. There is a high quality threshold for usefulness, as many of the minority language speakers are bilingual and are likely to prefer a foreign language version over a poor-quality translation into their native language. There is also a risk that well-intentioned but unaware users, without language ability in the minority language, inadvertently poison online resources or corpora with low-quality machine translation output. Such users may e.g. want to increase the coverage of resources such as Wikipedia, or may be legally required to produce texts in the minority language. In addition to being a nuisance to native speakers, the low-quality machine translation output can be problematic for future natural language processing systems based on machine learning, when it ends up in training data.

Many machine translation challenges remain, especially for low-resource languages with rich morphology. In the next section, I discuss how this thesis attempts to address some of these challenges.

1.1 Scope and contributions of the thesis

Table 1.1 summarizes the content of the Publications included in this thesis, grouped by recurring themes. Figure 1.1 shows the wide range of languages used in the experiments. Languages from diverse language families are included, increasing the convincingness of the results. All the languages are European. Most of the languages are morphologically complex.

The work in this thesis is motivated by a desire to improve translation into ***morphologically rich*** languages (MRL), especially in ***low-resource*** settings. The

		Subword segmentation						Low-resource MT			
		Approach			Learning setup						
Task	*Publication*	*Consistency*	*SWR*	*Cross-lingual*	*Unsupervised*	*Semi-supervised*	*Active learning*	*Cross-lingual*	*Data augmentation*	*Multi-task learning*	*Evaluation measures*
Morphological segmentation	I	intra	–	–	✓	✓	–	–	–	–	–
	II	intra	–	–	–	✓	–	–	–	–	–
	III	intra	–	–	–	✓	✓	–	–	–	–
	IV	intra	–	–	–	✓	✓	–	–	–	–
	V	–	✓	–	✓	–	–	–	–	–	–
Machine translation	VI	intra	–	✓	(✓)	–	–	–	–	–	–
	VII	src–trg	–	✓	✓	–	–	–	–	–	–
	VIII	trg–trg	–	✓	✓	–	–	✓	–	–	–
	IX	–	✓	–	✓	–	–	✓	✓	✓	–
	X	src–trg	–	–	–	–	–	–	–	✓	–
	XI	–	–	–	–	–	–	✓	✓	✓	–
	XII	–	–	–	–	–	–	–	–	–	✓

Table 1.1. Publications categorized according to themes. SWR is short for subword regularization. Consistency type abbreviations: intra = language-internal, src–trg = source–target, trg–trg = target–target.

task is framed as a ***one-to-many*** (Luong, 2016) setting in multilingual neural machine translation (MNMT), meaning that there is a single source language, but the possibility to exploit cross-lingual transfer between multiple (related) target languages. The task is called ***asymmetric-resource one-to-many*** translation, as the target languages are assumed to have very different amounts of training resources available. The low-resource morphologically rich language is on the target side of the translation. This direction receives much less research attention than the opposite, with English notably popular as the target language.

The Publications can be divided into those on the task of morphological segmentation (Publication I to V) and those on the task of machine translation (MT) (Publication VI to XII). This division primarily reflects how the methods are *evaluated*: the correctness of morph boundaries and the quality of translation, respectively. There are similarities between the groups: the overarching goal for the first group has been to develop methods for MT vocabulary construction, and Publications in the latter group may also present new segmentation methods.

In this work, the main approach to achieve the goal of better translation into MRL languages is through improving ***subword vocabulary construction*** using ***morphological segmentation***. For more on the task of morphological segmen-

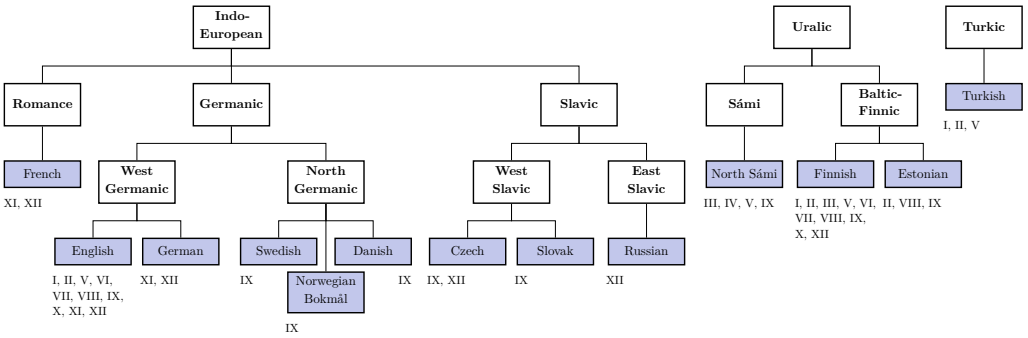


Figure 1.1. Partial language tree, showing the relations of the languages used in this thesis. The Roman numbers under the languages refer to the Publications in which the language is used.

tation, see Section 4.1. I begin with methods increasing the consistency of segmentation, later transitioning to methods embracing the ambiguity of segmentation as a source of noise for regularization. Three types of consistency in segmentation for multilingual translation are identified: language-internal, source–target, and target–target consistency. Language-internal consistency is so named because it concerns only a single language, while the two other consistency types apply to pairs of languages in multilingual settings.

Language-internal consistency means segmenting into units from a compact vocabulary of maximally productive subwords that can be easily combined into different words. Segmenting into linguistic morphemes is one way to achieve language-internal consistency, but it is also possible to intentionally undersegment frequently co-occurring sequences of affixes, or oversegment stems, as long as the decisions are consistent between words.

Publications I and II propose an extension of the existing Morfessor segmentation tool—called Morfessor FlatCat—in order to improve language-internal consistency using ***unsupervised*** and ***semi-supervised*** training. Morfessor FlatCat applies a Hidden Markov model morphotactics to avoid using valid morphemes in invalid contexts (e.g. segmenting the suffix *"s"* from the beginning of *"swing"*). Publication III[1], and Publication IV extend this work to a low-resource setting by applying ***active learning***. Publication VI combines rule-based and unsupervised morphological segmentation into a hybrid method.

Cross-lingual consistency increases symmetry between languages. ***Source–target consistency*** is most evident when the translated word is identical between source and target, making it possible to simply copy. However, the involved words are typically either rare or not present at all in the training data, i.e. out-of-vocabulary. Reasons for the scarce observations include rareness, novelty, or specific domain of the word. Representations based on word-level statistics for these difficult words are not reliable. While a special copy mechanism can over-

[1]And also a preliminary work (Grönroos et al., 2015) not included in the thesis.

come this challenge in word-level models, subword models are able to achieve copying without additional mechanisms. A rare word will be segmented into short, high-frequency subwords. If the identical word is segmented the same way in source and target, copying can proceed subword by subword.

Publication VII and VI tune Morfessor FlatCat towards matching segmentation granularity between source and target languages. Publication X uses an ad-hoc step of segmenting proper names into characters to allow the character level of a hybrid word/character decoder to attend to individual source characters.

When training a one-to-many multilingual model, ***target–target consistency*** arises between related target languages. Making cross-lingually consistent segmentation decisions increases the use of subwords with similar string representations and meanings, whether they occur in cognate words or elsewhere. Cognate Morfessor, proposed in Publication VIII, exploits automatically extracted cognates to achieve a cross-lingually consistent segmentation.

Cross-lingual transfer is exploited both in subword segmentation and in machine translation. In subword segmentation, moving from monolingual to multilingual models improves segmentation consistency.[2] In machine translation, moving from models for a single language pair to multilingual models is motivated by cross-lingual transfer, the regularizing effect of multi-task learning, and raising the level of abstraction of the internal representations. One long-term goal of this line of research is ***universal machine translation***, in which a single translation system is able to translate from any source language to any target language. Another goal is the quest for an ***interlingual meaning representation***, which abstracts away anything specific to any particular subset of languages. A third goal is ***low-resource translation***. In this work I pursue multilingual translation solely as a means to improve the quality of translation into a particular low-resource language. In this setting, using multilingual methods allows exploiting the large resources of a related language. For more on the scenarios for using multilingual machine translation, see Section 5.3.3.

In Publications VI and VII the cross-lingual learning is restricted to the tuning procedure. In Publication XI multilingual NMT training is applied. Publication VIII applies cross-lingual learning in two ways: as part of the proposed Cognate Morfessor method and through multilingual NMT training. Publication IX takes the importance of cross-lingual transfer further, with a highly asymmetrically resourced multilingual machine translation task.

Consistency of segmentation is motivated when two criteria are met: each word type is always segmented the same way, and the embeddings are the primary means for the cross-lingual transfer. An alternative approach relies on the ability of deep encoders to use context for disambiguating tokens. By explicitly generating noisy alternative segmentations during training, the model can be trained to be robust to segmentation ambiguity.

Publication IX exploits both monolingual and parallel data using three types of

[2]For the purposes of Table 1.1, joint segmentation in which training corpora are simply concatenated to train a single model are not considered to be a cross-lingual method.

auxiliary tasks: (i) subword regularization is applied to both types of data, (ii) a denoising sequence autoencoder allows training on monolingual data, and (iii) back-translation is used to convert monolingual data into synthetic parallel data. Using subword regularization on the parallel data is a form of ***data augmentation***. In Publication XI the task of multimodal translation requires data with three components: image, source language text, and target language text. Auxiliary data sets containing only two of these are augmented with synthetic data for the missing component. Publication X enriches the data with labels from morphological analysis, and exploits the additional labels using a target-side ***multi-task learning*** approach.

As a contribution to the evaluation of machine translation into morphologically rich languages, Publication XII proposes a new ***evaluation measure***.

1.2 Research questions

In later chapters of the thesis, the research questions will be referred back to using the number in parenthesis.

Concerning the topic of optimizing subword vocabulary construction for translation into morphologically rich target languages, the following research questions arise:

(RQ1.1) What is a suitable granularity of subword segmentation for the low-resource task? What should the total vocabulary size budget be?

(RQ1.2) Once a vocabulary budget is given, how should it optimally be used? How should the distribution of subwords look? Does it matter what data-driven segmentation method is used? Should one focus on improving the consistency of segmentation, or embrace variance in segmentation as a source of noise for regularization?

(RQ1.3) How can learning setups, such as semi-supervised learning and active learning, improve segmentation consistency and reduce the annotation effort in morphological segmentation?

(RQ1.4) Can cross-lingual transfer improve segmentation consistency?

Concerning the challenges posed by MRLs for evaluation of MT:

(RQ2.1) Can subword information improve evaluation of translation into morphologically rich languages?

Concerning which auxiliary tasks to use for machine translation:

(RQ3.1) When data is very scarce, is it better to train a small model on the low-resource data, or a larger model using also the auxiliary data?

(RQ3.2) Which types of auxiliary data are most useful for the low-resource task? Is cross-lingual transfer more useful than transfer from monolingual tasks?

(RQ3.3) How important is language relatedness for the cross-lingual transfer?

(RQ3.4) How to effectively exploit monolingual data? For which languages should one add monolingual auxiliary tasks?

(RQ3.5) Augmenting with synthetic data raises concerns of distributional mismatch between the natural and synthetic data. What is the effect of adding large amounts of synthetic data in multimodal translation?

(RQ3.6) How should learning of different tasks be scheduled? Is it better to train tasks sequentially one after the other, or all tasks in parallel at the same time? Can both sequential and parallel transfer be combined into a more effective schedule?

Concerning the quantity of data for machine translation training:

(RQ4.1) How does the amount of the data available for the low-resource language affect the translation quality?

1.3 Structure of the thesis

This thesis touches on a wide variety of topics in the fields of natural language processing and machine learning, including both statistical and neural machine learning systems for applications both in the areas of morphological segmentation and machine translation. This overview is intended as a presentation of both the necessary background and the most important parts of the contributions, readable as a coherent whole even without consulting the Publications. Some details of the experiments have been omitted in the interest of brevity.

The overview is divided into six chapters. After the introduction, there are two chapters providing the necessary background information on linguistics (Chapter 2) and machine learning (Chapter 3). The linguistic background focuses on the internal structure of words, and the machine learning background on the methods used in this work. A reader familiar with these fields will be able to skip over the chapters. When necessary, later chapters refer back to the relevant sections using footnotes.

The following two chapters on subword segmentation (Chapter 4) and machine translation (Chapter 5) concern the fields in which the contributions of this thesis belong. Both chapters begin with sections looking at the task, the relevant background, and prior work. The last section of these chapters present the novel ideas and primary results of the Publications. It is often difficult to draw a line between contributions to the two fields, as the subword segmentation methods are often applied to the task of machine translation. All the segmentation methods are placed in Chapter 4, but only the results of the intrinsic evaluations are placed there. Results of evaluating machine translation are found in Chapter 5.

Chapter 6 concludes the overview with a summary of the findings and a look at potential future work.

2. Linguistic background

> “Languages differ essentially in what they *must* convey, not in what they *may* convey.
> (Roman Jakobson 1959) ”

This chapter discusses the structure of language, focusing on morphology and the linguistic units of which words are composed. The computational approaches to morphology are discussed later in Chapter 4, once the prerequisite machine learning background has been presented in Chapter 3. In this chapter I also argue that suitable basic units should be selected specifically for each language and task.

2.1 The word

The term ***word*** is so central when speaking about language, that it can seem nearly indispensable. The definition of the word appears to be intuitive at first glance, but there are numerous subtle complications.[1]

Splitting sentences into words at white space may at first seem to work adequately for English. However, the approach is not language-universal, failing completely for languages such as Chinese that do not use spaces as separators. Even for English, it becomes immediately obvious that some punctuation also needs to be separated. Otherwise the *“English,”* with comma in the previous sentence would differ from an instance of *“English”* that is not followed by a comma.

Punctuation raises a number of issues concerning how to segment abbreviations (*“E.U.”*), contractions (*“shouldn’t”* vs. *“should not”*, *“inasmuch as”*), and symbols e.g. for currency (*“$9.99”*). A particularly complex example is a date range, e.g. *“1.3.–22.4.2019.”*, that could be considered anything between 1 and 11 words.

Hyphens are involved in many problematic cases, such as discoveries named after multiple people (*“Einstein-Rosen bridge”*) and loan words (*“vis-à-vis”*). Some hyphenated words are compounds, e.g. *“black-and-white”* which in Finnish corresponds to the closed compound *“mustavalkoinen”*. In English, compounds can in addition to hyphenation also be written in closed form (*“stingray”*) and open

[1] Brown et al. (1992) present a delightful account of some examples, in the form of a dialogue between Simplicio and Salviati, in a nod to Galileo Galilei.

form (*"manta ray"*).

Seeking an alternate definition, we might propose that a word must be possible to utter on its own. A prototypical instance of a word, e.g. *"run"*, clearly fits the definition: it can be used as a one-word command *"Run!"*. However, function words such as *"a"*, *"the"*, and *"of"* would be excluded by this definition, and many parts-of-speech would be left in unclear status.

Are these complications in the definition of the word of interest only to academics, or are there practical consequences? When building natural language processing applications, the choice of basic units is a practical decision with potentially severe effects on performance.

Linguistic annotations are also commonly produced at the granularity of words, e.g. the Berkeley FrameNet project (Baker et al., 1998), and Universal Dependencies (Universal Dependencies contributors, 2017). The decision is not universal, e.g. The Parallel Meaning Bank (Abzianidze et al., 2017) treats multi-word constituents as single tokens, and decomposes compositional compounds.

One of the fathers of modern linguistics, Ferdinand de Saussure uses the ***linguistic sign*** instead of the word as the central unit in his *Course in general linguistics* (de Saussure, 1916). The sign unites a ***signifier*** (a "sound image") with a ***signified*** concept. The signifiers can sometimes correspond to words, but also other units. For de Saussure, different inflected forms are different signs, but the same word. This notion of word is also called a ***lexeme***. The lexeme links a set of word forms to their meanings. The lexeme is represented by its dictionary form or ***lemma***. Dictionaries typically only index a single base form of each lexeme, meaning that inflected forms cannot be directly looked up.

Several lexemes sharing the same lemma are called ***homonyms***. For example the lemma *"bank"* can refer both to a slope beside a body of water (*"they pulled the canoe up on the bank"*) and a financial institution (*"the investment banks were responsible for the financial crisis"*). These can be distinguished e.g. using sense keys from the lexical database WordNet (Princeton University, 2010), in these examples bank%1:17:01:: and bank%1:14:00:: respectively.

The set of symbols used in language is ever expanding. New signs are coined through the process of ***word formation***. Signs called ***morphemes*** are the minimal units of meaning or grammatical function. In word formation, morphemes are composed together to form new signs. Morphological processes can be divided into two categories: inflection and derivation. ***Inflection*** describes how words adapt their form to their context. The available categories depend on the language and the part-of-speech. Some examples are number, case, person, mood, and tense. Inflection does not change the part-of-speech of the word, and the change in meaning is relatively regular.

Derivation alters the meaning of the word more substantially, for example the prefix *"un+"* in *"unreasonable"* inverts the meaning of *"reasonable"*. The latter also shows how derivations can be chained, as *"reasonable"* is already a derivation of *"reason"*. The latter derivation is an example of how derivation can change the part-of-speech: the suffix *"+able"* turns a noun into an adjective. ***Compound-***

ing is a type of derivation in which parts with the potential to stand freely are merged into a compound. Some languages use open compounds, i.e. compounds are written with a separating space between the parts. Other languages join the compound parts, sometimes adding hyphens or linking morphemes, or applying phonetic changes. This typological distinction is not strictly binary, e.g. English generally prefers open compounds, but exceptions, such as *"bookstore"* and *"family-owned"*, are common. Hyphenation is also used with some prefixes, e.g. *"semi-supervised"*.

Meaning. According to de Saussure (1916), the ***value*** (valeur) or meaning of signs only arises in their relation to other signs. It is not meaningful to speak of the value of a sign in isolation. As an example, in English *"sheep"* refers to the animal, while its meat is called *"mutton"*. In French, both concepts are referred to by the same word *"mouton"*. Even though *"mouton"* and *"mutton"* are ***cognate***, i.e. share an etymological origin, their value differs due to the differences in the way the semantic space is subdivided into concepts. Value is thus related to ***translational ambiguity***, in which a single concept in a source language is translated into multiple different concepts in the target language, depending on the context.

De Saussure divides the relations between signs into two categories: syntagmatic and paradigmatic. Signs in a ***syntagm*** are found co-occurring with each other, often in a particular order or position relative to each other. Signs in a ***paradigm*** occur in shared contexts, but are often mutually exclusive in a single example.

The sign is arbitrary, meaning that there is no way to determine the meaning from the apparent properties of the signifier, i.e. how it is written or pronounced. One exception is the class of onomatopoeic words, e.g *"bang!"* These words describe a sound, and are chosen so that they mimic it.

In general there is no such resemblance, and indeed words with similar appearance can have completely different meaning, e.g. *"porpoise"* and *"purpose"*. The mapping from symbols to meanings must therefore be learned from how they are used, or more specifically from the contexts in which they appear. This idea was restated by John Rupert Firth (1957) in his well known distributional hypothesis

> “ You shall know a word by the company it keeps. ”

Word-internal structure. According to Zellig Harris (1955), the distributional hypothesis applies to morphemes as well as words. Language is not only symbolic, but also ***compositional***. New larger symbols can be constructed from sequences of smaller symbols. A distinction can be drawn between meaning-bearing and non-meaning-bearing symbols. The arbitrariness of the sign implies that the smallest units, characters and phonemes, are not meaning-bearing.

Figure 2.1 shows a hierarchy of linguistic units. To start with the smallest units of written language, ***letters*** do not carry meaning in alphabetic languages. Se-

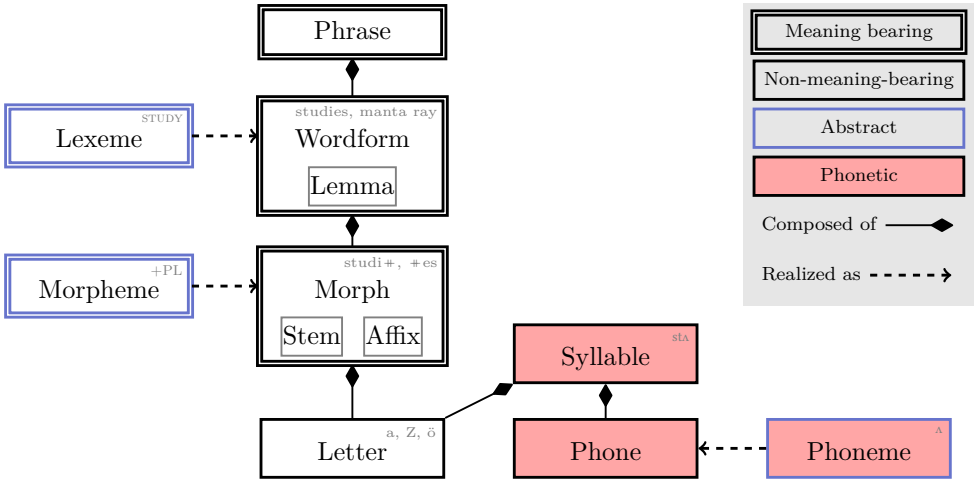


Figure 2.1. Linguistic units of varying sizes.

quences of letters can of course carry meaning, but the individual letters do not.[2] In a similar way as ***morphs*** are concrete surface realizations of abstract ***morphemes***, in the field of phonetics, ***phones*** are concrete realizations of ***phonemes***. ***Syllables*** are units of prosody, defined by the pronunciation. They do not carry meaning.

Morphemes are divided into stems and affixes. A prototypical ***stem*** is the substring of the lemma shared by all inflections of a word. In case of stem allomorphy, the stem may undergo some changes in certain inflections. A lexeme can thus have several stems. ***Affixes*** are further divided into prefixes, suffixes, infixes, and circumfixes. The term ***subword*** is used in natural language processing (NLP) for any concrete units that are smaller than words, including syllables and morphs, and sometimes characters. The use of this term is typically an indication that there is no emphasis on producing meaning-carrying units, although some subwords may correspond to meaning-carrying units by chance.

When composing meaning-bearing units into larger meaning-bearing units, the degree of compositionality may vary. In some cases the meaning of the whole is easily derived from the meaning of the parts, but in other cases the meaning of the whole is, to some degree, non-compositional. A non-compositional unit may be more appropriate to model as a whole, rather than subdividing it into the parts that it appears to consist of. When the meaning of an expression becomes sufficiently non-compositional, it is lexicalized, or turned into a new lexical unit.

For example Middle English *"hamlet"* could be analyzed through derivational morphology as its stem, from Old French *"hamel"* (little village) is cognate to Middle English *"hām"* (home) (Harper, 2019). While the diminutive suffix *"+let"* is still identifiable, the morpheme *"ham"* is in modern use found mainly in names. Therefore, it may be preferable for practical purposes to analyze the word as a single unit instead. In general, the optimal segmentation for an application may not coincide with the linguistically most correct segmentation.

[2] ***Characters*** extend the class of letters to symbols other than the alphabetic ones.

Compositionality of meaning applies also to units larger than words. Most phrases have a large degree of compositionality. ***Idioms***, e.g. *"bite the bullet"*, are an exception, being completely non-compositional.

Sentence level structure. A universal property of natural language[3] is that in order to communicate a message, it must be encoded as a linear sequence of symbols. When communicating in speech, the sequential nature is imposed by the passage of time. Written language also has a specific order in which the symbols should be read.[4]

The symbols in the sequence encode both concepts and the relations that connect the concepts to each other. The connecting relations consist of both the roles that the concepts play and the direction of the relation. There are two distinct ways in which this information can be encoded in the sequence: One way uses the choice of symbols, i.e. the paradigmatic relations of the signs. The other way uses the order in which the chosen symbols are placed, i.e. their syntagmatic relations.

Languages vary in how they make use of these two ways of encoding information. Languages with a ***strict word order***, such as German and English, make a clear distinction between the two by using symbol choice to encode concepts and word order to encode the relations. In languages with more ***flexible word order***, such as Finnish and Latin, both types of information are encoded by symbol choice. This is accomplished by introducing new symbols, morphemes, which are attached to the symbols encoding the concepts. Inflecting for case is an example of marking relations through morphology. As long as the attachments are not broken, the large-scale structure of the sequence can be reordered. The reordering can cause a change of emphasis, but does not substantially alter the meaning.

For example, in the English sentence *"The dog chases the cat in the basement"*, the order of *"dog"* and *"cat"* determines which is the subject and which the object. One translation of the sentence into Finnish is *"Koira jahtaa kissaa kellarissa."* However, the word order in the Finnish sentence can be modified radically without changing the meaning of the sentence, e.g. *"Kellarissa kissaa jahtaa koira."* The cat *"kissa"* is marked as the object by the partitive case, not by its position. The location is indicated by the function word *"in"* in English, but by the inessive case *"+ssa"* in Finnish.

The grammar of a language describes the rules for which roles must be filled, and in which order. These rules vary between languages. Grammar is divided into two parts—morphology and syntax—at the level of words. Morphology and syntax are similar to each other, in that both consist of lexical units and rules for combining them, and both contain productive aspects. Di Sciullo and Williams (1987) argue that the boundary between the two leaks to some extent, with the word being a bottleneck in the passage of information from the morphological to

[3]In constructed languages, the Unker Non-Linear Writing System (UNLWS) (Fink and Sai, 2002) is an interesting exception.

[4]While detours through references such as footnotes are possible in writing, the reader must choose to either embark on the detour or continue with the main sequence.

the syntactic.

Regarding units larger than words, a ***phrase*** is a group of words that function as a constituent in a sentence. All the dependents of the words in a phrase must be included in the phrase. It should be noted that in machine learning, a statistical phrase is any (frequently occurring) sequence of words, without regard for constituency.

Distributions of linguistic units. The choice of basic units has a striking effect on the observed distribution. For alphabetic languages, the number of distinct letters is small (typically 20–40), and any document of reasonable length is likely to contain all of them. The number of words on the other hand is unbounded, and new ones continue to be encountered even after observing a substantial corpus. A famous observation called Zipf's law (Zipf, 1932),[5] states that the number of occurrences of words tend to follow a ***power-law distribution***: the k^{th} word in a list sorted by frequency has a frequency inversely proportional to k. One consequence is that there is a long tail of rare words. The distributions of subwords lie somewhere between these two extremes, with the particular shape of the distribution defined by the method of segmentation.

Languages vary in the granularity of words, as a result of different balance between use of symbol choice and order for encoding information, and also as a result of orthographic differences. Therefore I argue that rather than defaulting to words, suitable basic units should be selected for each language and task. The word is an end result of linguistic analysis, rather than its starting point.

2.2 Linguistic typology

Linguistic typology characterizes and classifies languages according to their structural and functional patterns. Recently linguistic typology has been approached also by means of machine learning (Östling and Tiedemann, 2017a; Malaviya et al., 2017; Asgari and Schütze, 2017). While there are several typological aspects that can be considered, three of them will be discussed here: word order, degree of synthesis, and degree of fusion.

The primary ***word order*** typology concerns the order of subjects (S), objects (O), and verbs (V). The most common orders are SOV and SVO (Dryer, 2013). The ordering of other sentence constituents, such as adjective–noun pairs, can also vary between languages. Languages with free word order tend to use morphological markers, e.g. inflecting for case (McFadden, 2003).

The ***degree of synthesis*** concerns the degree to which the atomic units of sentences are composed into longer, more complex words. ***Isolating*** and ***analytic*** languages have a preference for freestanding words. Grammatical relations are marked by word order or by parts-of-speech such as prepositions. This results in a low morpheme-to-word ratio. Chinese, Vietnamese, and English are exam-

[5] Illustrated in Figure 5.6 on page 137.

ples of analytic languages. ***Synthetic*** languages have a high morpheme-to-word ratio. Grammatical functions are marked through morphology. Morphology is not limited to function categories, but can also express content. ***Polysynthetic*** languages, such as Inuit, use morphology to such a large extent that individual words can function as complete utterances. Languages with very high morpheme-to-word ratios are called ***morphologically rich languages*** (MRL).

The ***degree of fusion*** describes how much morphs are transformed by interaction with other morphs. In ***agglutinative*** or concatenative morphology, morphemes remain unchanged when concatenated into a word. In ***fusional*** morphology, consecutive morphemes become fused together. Alternatively, one morpheme in a fusional language can be considered to mark several morphological features, also called ***formative exponence*** (Bickel and Nichols, 2013). ***Introflective*** morphology refers to inflections marked by stem change that is not easily separable into concatenated morphemes, seen e.g. in irregular inflections such as *"sing–sang"*. Morphological segmentation is better suited for agglutinative languages, as fusion can make the exact location of morph boundaries fuzzy.

It should be noted that no language is completely of one type. While entire languages are often classified based on their predominant type, it is more precise to refer to the types of individual phenomena within a language. For example in German, five different types of morphological phenomena can be observed:

isolating:	*"ich werde machen"* (I will do)
agglutinative:	*"ziehen"* (pull),
	"anziehen" (put on, dress),
	"miteinbeziehen" (include)
fusional:	*"Kindes"* (of the child)
	(in which the ending -es marks number, gender, and case)
introflective:	*"tragen"* (carry) – *"trug"* (carried),
	"Mutter" (mother) – *"Mütter"* (mothers)
compounding[6]:	*"Kleinstadt"* (small town), *"blaugrün"* (blue-green),
	"Fleisch-fresser" (carnivore)

(From Skalička (1979), reproduced by Igartua (2015))

Bound and free morphemes. A free morpheme can stand on its own as a word. A bound morpheme must always be attached to another morpheme. There is always at least one free morpheme in a word, but there can also be multiple. The stems are free morphemes, while the affixes are bound. Generally compounds are formed by combining several free morphemes, but there are exceptions, such as *"cranberry"*, where the fossilized term *"*cran"* cannot stand alone.

Open and closed classes. There are many ways that lexical units can be divided into classes based on their usage, such as part-of-speech classes. Regardless of

[6]Skalička (1979) call this type polysynthetic, but I would not use the term for simple compounding, reserving it for more complex synthesis involving e.g. incorporation between verbs and nouns.

which classification is used, the classes can be described using the property of openness. New words are created as needed when novel concepts in the world need describing. An ***open class*** admits new members when language is extended, while a ***closed class*** can be enumerated exhaustively. In practice, the distinction is not strictly categorical: classes can be more or less likely to gain new members. As an example of extending a class from the closed end of the spectrum, consider the proposed gender neutral pronouns such as *"zie"*.

Closed classes tend to express grammatical function, e.g. the parts-of-speech articles, pronouns, and prepositions, or the morphemes inflecting for number, gender, and case. Open classes carry more of the semantic information. Examples include content word parts-of-speech such as nouns, verbs, and adjectives, the stem and prefix morpheme classes, and the class of compound words. Proper names, loan words, technical terms, and portmanteau words (e.g. *"brexit"* from *"Britain"* and *"exit"*) are some examples of open classes of words frequently gaining new members.

Even the meaning of nonce words, previously unseen words coined for a specific purpose, is understandable, based partly on inference using the single sentence context they are seen in and partly based on the morphology of the word (Štekauer, 2002).

Allomorphy is a phenomenon in which a single abstract morpheme can be represented with several different surface morphs. The allomorphs have the same meaning but different surface strings. E.g. in Finnish, vowel harmony determines whether the inessive is marked by *"-ssa"* or *"-ssä"*. *"Talossa"* contains back vowels in the stem, while *"kylässä"* contains front vowels. While the difference between the allomorphs is phonetic, not semantic, the allomorphs nevertheless have distinct context distributions. Allomorphy is not restricted to affixes, but introflective changes such as vowel and consonant alternation also affect stems.

2.3 Theories of morphology

There are three well-known approaches to modeling morphology: Word-and-paradigm, Item-and-arrangement, and Item-and-process (Hockett, 1954).

The oldest of the three, ***Word-and-paradigm*** (WP), only addresses inflectional morphology. It describes a paradigm using two components: an inflection table and a list of lexemes belonging to the paradigm. The inflection table shows the paradigmatic relations of all inflected forms of a prototypical lemma. All lexemes in the list can be inflected by analogy with the table. The analogy must be exact; even a single-letter change necessitates defining a new (sub)paradigm. For example *"play"* and *"enter"* belong to the same paradigm: *"play–plays–playing–played–played"* and *"enter–enters–entering–entered–entered"*.

Item-and-process (IP) describes morphology as a set of processes that transform one (or several) item(s) into a new item. For example, we may describe a

process deriving nouns from verbs as

$$[\mathrm{X}]_{\mathrm{Verb}} \rightarrow [\mathrm{X} + \mathrm{er}]_{\mathrm{Noun}}.$$

While the example only appends a suffix, processes can perform arbitrarily complex transformations, e.g. the vowel replacement necessary for transforming *"take"* to *"took"*. It should be noted that the process describes a transformation in configuration, not in time or historical development. Nevertheless, IP places word forms into a directed graph, with certain forms being more central than others.

Item-and-arrangement (IA) in its core describes words as a set of morphs and their arrangement. The arrangement is restricted by morphotactics, describing which morphs may occur together and the order in which they must be placed. IA is well suited for languages using agglutinative morphology, in which morphs are concatenated together without change. Allomorphy, fusion, and other forms of irregularity pose challenges for IA. If phonological rules require a morph to be replaced with an allomorph due to the presence of certain phonemes in other items, the interaction of morphs is not limited to the arrangement. To address allomorphy, abstract morphemes can be used as the items, with an additional allomorph selection step given the arrangement.

Irregular forms pose a challenge even under this revised model. For example the *"take–took"* inflection has no identifiable suffix marking the +PAST morpheme. Some of the possible solutions (Hockett, 1954) include analyzing *"took"* as

1. a single morpheme TOOK, distinct from TAKE,
2. a portmanteau morph combining TAKE and +PAST,
3. an allomorph of TAKE in sequence with a null morph as an allomorph of +PAST,
4. a discontinuous allomorph *"t-k-"* of TAKE and an infixed allomorph *"-oo-"* of +PAST, or
5. (limited to pronunciation) as *"take"* and a replacive morph /ʊ/ ← /eɪ/.

Only the fourth option is acceptable according to Hockett (1954). He considers the first unacceptable, as the analysis of the irregular form is not parallel with regular past tense forms. The second and third he considers to be arbitrary: Why should not regular forms be analyzed as portmanteaus? Why should +PAST become the null morph rather than TAKE? The fifth is inconsistent with the notion that morphs should be composed of phonemic material.

Methods for morphological segmentation, such as the ones proposed in this thesis, rely implicitly or explicitly on the IA model.

2.4 Relations between languages

Languages are constantly but slowly evolving. To some extent this is due to a changing language use environment necessitating innovation of new words, ***ne-***

ologisms. Some of these will propagate from one language to another in the form of ***loan words***. Another contributing factor is the fact that for a language to be carried into the future, it must be learned by new generations of language learners, some of which are already speakers of another language.

The field of ***historical linguistics*** is concerned with the evolution of languages and their relations. Languages are classified into families by finding systematic similarities. Within each family, languages are grouped into subfamilies if they share significant innovations. For a partial language tree covering the languages used in this thesis, see Figure 1.1.

In the case of the Romance language family, historical linguistics benefits from the parent language (Latin) having very extensive records. In cases where available records are scarce, or even if the parent language is unattested, the comparative method can be used to reconstruct the parent language. Reconstruction is achieved by comparing cognates of the daughter languages and undoing the regular sound changes. Protolanguages with attempts at reconstruction this way include Proto-West-Germanic and Proto-Indo-European (Ringe, 2017).

Cognates are the primary tool for the comparative method in historical linguistics. Cognates are words in different languages with a shared etymological origin. An ideal cognate pair is similar in three ways: orthographically (i.e. represented as identical or nearly identical strings), semantically (i.e. refer to the same or similar concepts), and distributionally (i.e. are used with similar frequency in similar contexts). The word cognate comes from the Latin, meaning "born together". Care must be taken with words with similar string representation but different semantics, called "false friends". E.g. Finnish *"vaimo"* (wife) should not be translated as Estonian *"vaim"* (spirit, ghost).

Cognates are useful in multiple ways also outside of historical linguistics. Cognates have been identified as easy vocabulary items helpful for learning a second language (de Groot et al., 1994).

For the comparative method to be effective, proper names and loan words are typically excluded. In natural language processing it is common to use a broader definition of the term cognate (e.g. Kondrak, 2001), accepting proper names and loan words as cognates. With the broad definition, names of persons, locations, and other proper names yield cognates for any language pair written in the same alphabet. However, for many language pairs, these broad cognates are not inflected. Cognates are more abundant in related languages, and when the languages are also morphologically rich, it becomes necessary to model the similarities of inflection between the cognates.

The subtask of cognate extraction has seen much research effort (Mitkov et al., 2007; Bloodgood and Strauss, 2017; Ciobanu and Dinu, 2014). Most methods are supervised, and/or require rich features. Some work on cognate identification from the perspective of historical linguistics includes Rama (2016) and Kondrak (2009). The aim is to determine whether cognate candidates truly share an etymological origin or are merely superficially similar.

3. Machine learning background

> “All models are wrong, but some are useful
> (George Box, 1976) ”

Machine learning (ML) is a field of science that studies how to program machines to modify their behavior based on experience learned from observing data. The programmer supplies prior information by selecting the model family, the algorithm for learning the parameters, and possibly some hyper-parameters. The concrete model is selected from the model family by fitting it to some training data. This differs from traditional programming, in which the behavior of the machine is fully determined by the programmer.

Artificial intelligence (AI) is a related concept, describing the creation of artificial systems that exhibit some characteristics of intelligence. As intelligence is difficult to define, AI is a vague and moving target, often referring to tasks in which machines are not yet proficient. In the early days, AI was commonly built by hand-crafting rules, but currently it is typically implemented through machine learning. AI in which logical rules are used for manipulating symbols is sometimes referred to as Good Old-Fashioned Artificial Intelligence (GOFAI). Both the objects and their relations are described as discrete symbols, which are collected into a knowledge base. The dependence on a knowledge base makes the approach labor-intensive and brittle. Rule-based AI is still used in some niche applications, such as controlling opponents in computer games.

Statistical and probabilistic methods are applied in machine learning. ML can be contrasted against statistics in general, by considering how the end result is used. In traditional statistics, humans gain insight from examining the optimal model parameters directly. In ML, the learned model is typically applied to some new data in an automatic or semi-automatic way, to make a prediction or decision. Huge, overparameterized models are undesirable in statistics, as they are too large to be easily interpreted. In machine learning, such black box models are more acceptable.

ML has enabled systems that come a long way towards solving complex tasks that are difficult to understand and therefore difficult for programmers to write rules for. One characteristic of ML systems is the ability to learn behavior that was not intended by the programmer. This can be both a strength and a weakness. It is a strength when the system learns from experience to behave correctly in a situation that was not foreseen by the programmer, but was either present in the training data or could be inferred from related situations. It can be a weakness

when it is important for the system behavior to be very predictable.

The following chapter presents the machine learning background necessary to understand this thesis. The chapter is based on several textbooks (Manning and Schütze, 1999; Bishop, 2006; Koller et al., 2007; Alpaydin, 2010; Russell and Norvig, 2010; Goodfellow et al., 2016). The focus of the chapter lies on parametric machine learning. A parametric machine learning method requires three components: a model, a loss function, and a learning algorithm. I will start by introducing the types of tasks that can be solved using machine learning, then describe how to construct models, and finally how to fit the models to data using the loss and the learning algorithm.

3.1 Types of machine learning tasks

In ***classification*** the aim is to assign to each input a label from a finite set of discrete labels. ***Regression*** instead outputs one or more real numbers.

$$\text{Classification} \quad f : \mathbb{R}^n \mapsto \{1, \ldots, V\}; \qquad \text{Regression} \quad f : \mathbb{R}^n \mapsto \mathbb{R} \tag{3.1}$$

Clustering is an unsupervised task, in which similar examples are grouped together. The groups are identified by cluster labels, but they are not meaningful like the classification labels. The clustering is considered equivalent even if the cluster labels are permuted, as long as the same examples are clustered together. In order to turn the cluster labels into the same kind of labels as in classification, additional analysis is needed to find a mapping from cluster labels to meaningful labels. For example, even if the clustering algorithm has placed *"f"* and *"v"* in the same cluster, it still requires additional information to identify them as being labiodental fricatives.

In ***sequence generation***, the output is a vector of predictions $\boldsymbol{t} = (t_1, \ldots, t_J)$. Sequence generation can be implemented as repeated autoregressive classification. In each timestep, the current decoder state is used as input to a classifier, which has as its labels the symbols of the output vocabulary. The classifier label is emitted, and also fed back in to the network to advance the decoder state. In sequence generation, the output symbol is selected from a vocabulary with fixed-size $t_j \in \mathrm{L}; \quad |\mathrm{L}| = V$. The symbol is selected from a ***categorical distribution***, which is parameterized by a vector of V probabilities, summing to one.

When drawing a single sample, the categorical distribution and the related ***multinomial*** distribution are the same. On repeated draws, the categorical distribution yields an ordered sequence, while the multinomial yields the (unordered) counts of events instead. A categorical distribution can be produced from an unnormalized vector output using the ***softmax*** function

$$\mathrm{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{k=1}^{V} \exp(x_k)}. \tag{3.2}$$

3.2 Parametric modeling

Three components are needed for parametric machine learning: a model, a loss function, and a learning algorithm. The model distribution is ***parametric*** if it is defined by a predetermined number of parameters. The values for the parameters are determined through optimization, but their number does not change. The model capacity is fixed in advance.

Classic statistical parametric models typically use a small number of well-known distributions defined by a small number of parameters, e.g. Gaussian distributions. Also neural networks can be viewed as parametric models. Even though the number of parameters can be very large, and the values of the parameters are not interpretable as is the case with the well-known distributions, the number of parameters is nevertheless fixed.

The alternative is to have a ***nonparametric*** model, where parameters are added and removed as needed. The model has a flexible capacity. For example, if a subword lexicon of variable length is learned, and each subword has some parameters (e.g. counts) associated with it, adding a new subword expands the model by the parameters for the new subword.

As an extreme example of a nonparametric model, consider nearest-neighbor models. All examples in the entire training data are needed to determine the model distribution.

3.2.1 Graphical models

Generating a sequence of natural language is a multivariate prediction problem. A naïve solution divides the multivariate prediction problem into multiple univariate predictions, i.e. learns a per-position classifier $\boldsymbol{s} \mapsto \boldsymbol{t}_j$ and runs it $\boldsymbol{J}$ times independently. This approach is poorly suited to domains such as natural language, where output variables have strong and complex dependencies. For example, in English adjectives seldom follow nouns. It is therefore important to be able to predict the sentence as a whole, using ***structured prediction***.

Graphical models are a way to represent a complex multivariate distribution as a product of local factors on smaller subsets of variables. The vertices of the graph correspond to random variables, while the edges represent a direct influence between the connected random variables. The graph structure encodes a factorization of the probability density, or alternatively a set of conditional independence relationships present in the distribution. A good factorization results in a compact, easily learned parameterization.

Based on the form of the learned probability distribution, graphical models are divided into generative and discriminative models. The distinction has a counterpart in the graphical formalism used: it's typical to use directed graphs for generative models and undirected graphs for discriminative models.

In ***generative*** learning, the joint distribution $\boldsymbol{P(X,Y)}$ of features and labels is modeled. It is called generative learning because complete new data points

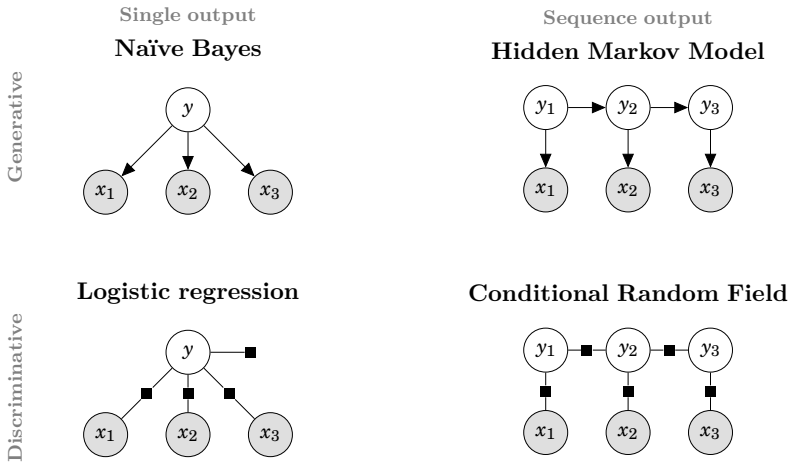


Figure 3.1. Relations between four graphical models.

can be generated by sampling from the joint distribution. A directed graphical model, presented as a directed acyclic graph (DAG), describes how a distribution factorizes into local conditional probability distributions. The direction of the arrows determines the ***generative story***: the order in which variables are drawn to generate a sample from the data. As the probability of the whole model is a product of conditional probabilities, it is properly normalized.

In ***discriminative*** training, the conditional probability of the labels given the features $\boldsymbol{P(Y \mid X)}$ is modeled directly. This distribution is the minimal needed for classification. In undirected models, the probability distribution is a product of factors Ψ. Each factor Ψ_a depends only on a subset of the variables $\boldsymbol{Y}_a$. The factor can be thought of as a measure of how compatible the assigned values $\boldsymbol{Y}_a$ are with each other. There is much freedom in defining the factors, as they don't need to be probability distributions: it is sufficient that they are non-negative scalars. As a downside, normalization is needed to ensure that the distribution sums to one. The normalization is achieved using a partition function $\boldsymbol{Z}$, that is a summation over exponentially many possible assignments to $\boldsymbol{Y}$. As such, $\boldsymbol{Z}$ is in general intractable, and much of the work on undirected models is focused on approximations for it.

A generative model must be able to represent all patterns in $\boldsymbol{X}$, regardless of their usefulness in predicting $\boldsymbol{Y}$. The features may be connected by complex dependencies, so constructing a probability distribution over them can be challenging and lead to intractable inference. Ignoring the dependencies simplifies the model, but can lead to reduced performance. Discriminative training is free to disregard any non-informative patterns, as it only cares about the mapping from $\boldsymbol{X}$ to $\boldsymbol{Y}$. Dependencies that involve only variables in $\boldsymbol{X}$ are simply ignored. Global structure is more important for generative models. For discrimination, it may be enough to recognize salient features, e.g. the nose and eyes for faces, while a generative model also needs to place them in the correct relation to each other.

The ***Markov property*** is a very useful simplifying assumption for sequence

modeling. For a sequence structure, a Markov model of degree n conditions the current prediction only on a fixed window of n previous variables. Given the fixed window of observations, the prediction is assumed to be independent of older history

$$P(t_j \mid t_{0:(j-1)}) \approx P(t_j \mid t_{(j-n):(j-1)}) = P(t_j \mid \mathrm{pa}(t_j)) \tag{3.3}$$

For a general structure, the local Markov assumption states that each variable t_j is independent of its nondescendants given its immediate parents $\mathrm{pa}(t_j)$.

Statistical ***language models*** (LM), such as the n-gram LM, make use of the Markov assumption to simplify estimation of parameters and to control sparsity. While early neural language models (Bengio et al., 2003) also made use of the Markov assumption, modern ones (Mikolov et al., 2010) typically do not, in order to model arbitrarily long dependencies.

The hidden Markov Model (HMM) is a latent variable generalization of the Markov model. The state sequence $y_1, \ldots, y_I$ is not observed, and must be inferred based on the observed features $x_1, \ldots, x_I$. Each state y_i emits an observation x_i from the conditional distribution $P(x|y)$. The transitions between states are determined by another conditional distribution $P(y_i|y_{i-1})$. The resulting joint probability of the observation sequence is

$$P(x_1, \ldots, x_I) = \prod_{i=1}^{I} P(y_i|y_{i-1})P(x_i|y_i). \tag{3.4}$$

The HMM generalizes the naïve Bayes classifier to sequence modeling. The standard inference algorithms for HMMs are the Viterbi, forward, and forward–backward algorithms. The Viterbi (1967) algorithm is a dynamic programming algorithm for computing the most probable state sequence for a given sequence of observations. The related forward algorithm instead computes the marginal probability of an observation sequence, accounting for all possible state sequences. The forward–backward algorithm, also known as the Baum-Welsh algorithm (Baum, 1972), is an application of the Expectation-Maximization algorithm[1] for estimating the parameters of an HMM when only observation sequences are available.

Conditional random fields (CRF) are probabilistic models for discriminative structured classification. They are expressed as undirected probabilistic graphical models. CRFs can be seen as generalizing the logistic regression classifier to structured outputs. In the ***linear-chain CRF***, the variables are arranged sequentially. CRFs can also be used with more general graphical structures. Linear-chain CRFs bear a structural resemblance to hidden Markov models, while relaxing the assumption of the observations being conditionally independent given the labels, which makes them usable with rich, partly redundant features. Figure 3.1 places the HMM and CRF models in a wider family of graphical models.

[1] See Section 3.5.1.

3.2.2 Neural models

Deep learning—the use of neural networks with many layers—has enjoyed great success since the 2010s, beginning when Krizhevsky et al. (2012) won the ImageNet object recognition challenge using a convolutional network. However, neural methods had a very long history before that, including applications to Natural Language Processing (NLP). Modern deep learning builds on a long tradition of work under many names: cybernetics in the 1940s-1960s, connectionism in the 1980s-1990s, and finally deep learning after 2006. Let us review some of that history.

The term ***artificial neural network*** (ANN) implies a perspective of biomimicry. Biological neurons acted as inspiration for the development of artificial neurons, and some work has been used to improve understanding of brain function. The nodes of the ANN act as analogs to neurons, while the weighted connections between the nodes act as synaptic junctions. The McCulloch and Pitts (1943) neuron was an early model of activity in the brain. It is a linear model capable of performing two-class classification, after a human operator has set the weights to achieve the desired function. Building systems that aim to replicate human or animal brains is no longer a central aim of current neural network research. Instead models are designed so that they perform well on tasks, regardless of biological plausibility.

The perceptron (Rosenblatt, 1958) was the first model capable of learning its weights when exposed to training data. The adaptive linear element (ADALINE) (Widrow and Hoff, 1960) was trained using an algorithm that is a special case of stochastic gradient descent (SGD). SGD and its variants are still the most commonly used training algorithms. The example task used by Widrow and Hoff (1960) was language related, namely a toy task of optical character recognition, classifying the characters T, G, and F using a 4-by-4 pixel grid.

The early systems had limited success, even though many of the necessary innovations had already been made: backpropagation (Rumelhart and McClelland, 1985), nonlinearities, stochastic gradient descent, and deep networks. Success was delayed for practical reasons. More computing power and memory enabled two kinds of breakthroughs: larger model sizes benefiting more from depth of representation, and wide-ranging experimentation with training procedures, leading to insights about proper initialization (Glorot and Bengio, 2010) and regularization[2] of neural networks. Large amounts of training data are generally required to reach high performance. Only recently have advances been made in methods for training neural networks with smaller data. The use of GPU computing to parallelize linear algebra computations (Steinkraus et al., 2005) and availability of programming libraries such as Theano (Bergstra et al., 2010; Bastien et al., 2012) and PyTorch (Paszke et al., 2017) have also contributed to the success of deep learning. Recently, techniques for training ever deeper networks have been

[2]See Section 3.3.4 for more on regularization.

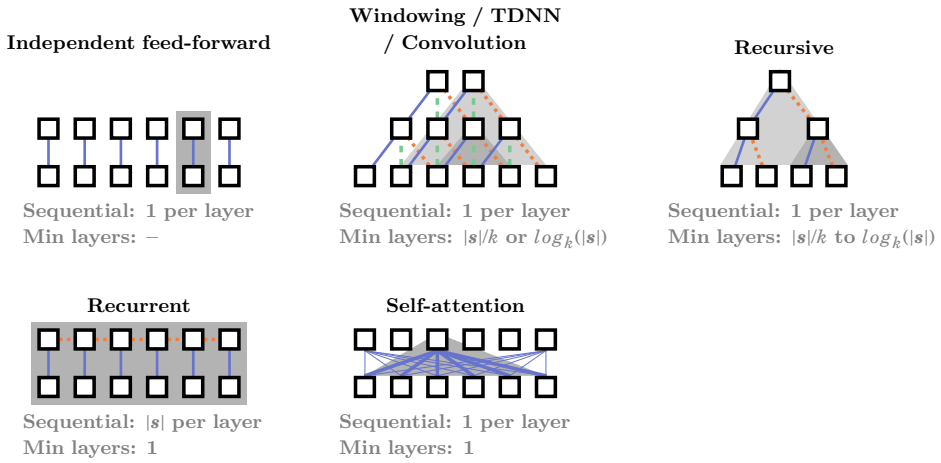


Figure 3.2. Approaches to sequence modeling in neural networks. The colors and dashing indicate weight sharing through time. *Sequential* indicates the number of operations per layer which must be performed in sequence, showing the potential for parallelization. *Min layers* shows the minimum number of layers needed to cover arbitrary dependencies in a sequence of length $|\boldsymbol{s}|$. k is the width of the convolution kernel or the arity of the tree. The shaded areas show the span covered by one (darker) and two (lighter) layers. Some connections for self-attention are omitted for clarity.

a driver for improved performance. These techniques include increasing effective minibatch size using gradient accumulation, increasing depth through model parallelism (Coates et al., 2013), low precision computation (Courbariaux et al., 2014), and improving gradient flow using transparent attention (Bapna et al., 2018).

Early methods achieved depth by combining unsupervised pretraining with supervised training (Bengio et al., 2007). The depth was added incrementally, by freezing previously trained layers and adding a new, deeper one to be trained. The use of unsupervised representation learning is related to the transfer learning[3] approaches that are still very popular, as evidenced e.g. by the wide use of contextual representations from pretrained BERT (Devlin et al., 2019) models in downstream NLP tasks.

Approaches to sequence modeling

A defining characteristic of natural language is that it is expressed as variable length sequences. This distinguishes natural language from domains such as image processing, where typically images of fixed pixel dimensions are used.

Figure 3.2 compares neural architectures for sequence modeling. The simplest neural network architecture is ***feed-forward*** (FF) networks, which have an input of a fixed size, and do not contain loops providing feed-back connections or more advanced forms of memory. The trivial way of processing with a FF network applies it to each timestep of the sequence separately. However, this unsatisfactory approach results in independent predictions for each timestep, with no sequence modeling.

[3]See Section 3.4.2 for more on transfer learning and multi-task learning.

To give the network some ability to use temporal information, a sliding window over the input can be used. As the width of the window is fixed, a FF network is applicable. The architecture is called Time Delay Neural Networks (TDNN) (Waibel et al., 1989), and it computes a 1-dimensional ***convolution*** over the time axis. 2-d convolutions are used in the image and spectral domains. Convolutional networks exploit temporal and spatial structure through parameter sharing: in each timestep the same weights are applied to the positions of the window. A single layer TDNN is similar to a continuous version of an n-gram, as both predict the probability of the i:th token from a fixed length history $i-n:i-1$. In deep convolutional networks, the deeper layers combine windows of features produced by the lower levels, which increases the span of inputs that are covered. A deep convolutional network is thus able to use more distant information than a shallow one. In order to connect all inputs of a sequence of length $|\boldsymbol{s}|$ to each output requires a stack of $|\boldsymbol{s}|/k$ convolutional layers with contiguous kernels of width k, or $\log_k(|\boldsymbol{s}|)$ in the case of dilated convolutions.

If the sequence represents the leaves of a known tree structure, e.g. a parse tree produced by an external parser, then the nodes of the neural network can be connected according to this structure. A ***recursive*** network can be used for composing a single vector by propagating information from the leaves to the root. All the compositions share parameters. Compositions in the same layer can be performed in parallel, but the minimum number of layers needed to compose the entire sequence depends on both the length of the sequence and the shape of the tree: logarithmic in the best case of a balanced tree, and proportional to the length of the sequence in the worst case of a left/right-branching tree.

A downside of recursive networks is that the structure of the tree cannot be learned during training. If we want to be able to learn a structure, a different approach is required. An elegant solution to this problem is provided by the ***gating mechanism***

$$\boldsymbol{y} = \sigma(\boldsymbol{W}\boldsymbol{c} + \boldsymbol{b}) \otimes \boldsymbol{x} \tag{3.5}$$

where $\boldsymbol{W}$ are the weights of the gate, $\boldsymbol{c}$ is the context determining the operation of the gate, $\boldsymbol{b}$ is the bias vector, and $\boldsymbol{x}$ is the input. The sigmoid σ outputs a value between zero and one. Multiplying the input elementwise with these numbers allows the gate easily to learn to either output the value unchanged (saturating to one) or suppress the output entirely (saturating to zero). Closing the gate effectively prunes the connection from the network. Neither the activation nor the gradient will pass through the closed gate. The learning starts from a fully connected network, with unnecessary connections softly pruned away. This type of pruning cannot reduce the number of parameters or the computational complexity. As the mechanism is fully differentiable, it in effect allows the network to learn the desired structure from the data.

Recurrent neural networks (RNN) (Elman, 1990) take a different approach to sequence modeling, conditioning the output on not only the input $\boldsymbol{s}_i$ but also

a representation of the entire preceding history encoded in a recurrence vector

$$\boldsymbol{h}_i = f(s_i, \boldsymbol{h}_{i-1}). \tag{3.6}$$

A single layer has access to the entire input, but processes it sequentially, which limits the use of parallelism for speeding up computation. While recurrence makes it theoretically possible to model arbitrarily long dependencies, in practice long paths lead to problems such as exploding and vanishing gradients.[4] In order to address these challenges, advanced recurrent units employing the gating mechanism have been introduced. These include the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Chung et al., 2014).

A single RNN reads the sequence in only one direction, which causes two problems. First, the intermediary representations at different timesteps contain very different amounts of information, from the fist timestep having seen only the first symbol to the last timestep having to represent the entire sequence. Second, the path length from the last timestep encoding the full sentence to the first input is very long, resulting in information loss and slow learning.

Schuster and Paliwal (1997) present the ***bidirectional*** recurrent network, which combines a forward network $\overrightarrow{\text{LSTM}}$ to read the sequence left-to-right, with a backward network $\overleftarrow{\text{LSTM}}$ reading right-to-left.[5] The outputs of the two networks are concatenated to form the final output. E.g. at index i, the output is the concatenation

$$\boldsymbol{h}_i = [\overrightarrow{\text{LSTM}}(s_{0:i}); \overleftarrow{\text{LSTM}}(s_{I:i})]. \tag{3.7}$$

If only the final states are used for encoding the whole sequence, the bidirectional network gives a modest benefit through the alternative backward reading. The bidirectional network is even more beneficial when the entire output is used as a variable-length encoding, e.g. as input to an attention mechanism. In a bidirectional network, each timestep of the output has access to information from the entire sequence.

The attention mechanism was introduced by Graves (2013), and applied to machine translation by Bahdanau et al. (2014). It is often useful to update a neural state $\boldsymbol{h}$ based on a variable length context sequence $\boldsymbol{V}$. The approaches presented previously have addressed the problem by iteratively building up a single vector that encodes the entire context. However, only a part of the context sequence is relevant to the current computation, and the relevant part varies between computations. The attention mechanism solves this problem by each time selecting anew the relevant parts of the attended context values $\boldsymbol{V}$, and summarizing them into a single output vector. The selection is performed based on a query vector $\boldsymbol{q}$, and the key matrix $\boldsymbol{K}$, which contains one entry for each of the values $\boldsymbol{V}$.

[4] See Section 3.5.2 for more.

[5] The backward direction is indicated by the reversed index $s_{I:i}$ in Eq 3.7

An attention distribution is formed as a function $f(\boldsymbol{q},\boldsymbol{K})$, normalized using the softmax function. The summarized context is given by the weighted average of the attended values $\boldsymbol{V}$, where weights are given by the attention distribution

$$\text{Attention}(\boldsymbol{q},\boldsymbol{K},\boldsymbol{V}) = \text{softmax}(f(\boldsymbol{q},\boldsymbol{K}))\boldsymbol{V}. \tag{3.8}$$

In the simplest case, f is simply the dot product $f(\boldsymbol{q},\boldsymbol{K}) = \boldsymbol{q}\boldsymbol{K}^T$.

Models for NLP problems involving a mapping from one sequence to another (seq2seq) typically employ encoder-decoder architectures. The ***encoder*** encodes the input into an intermediate representation, from which the ***decoder*** generates the output sequence.[6] The predicted attention weights are sometimes treated as a kind of soft word alignment between input and output. This is seen as particularly useful for visualizing the operation of e.g. a translation system. The explanatory power of attention weights has also met with criticism (Koehn and Knowles, 2017; Moradi et al., 2019). As encoders have the ability to incorporate information from other timesteps, and attention only needs to produce context relevant for the prediction, there is no requirement that attention corresponds to e.g. translational equivalents.

In the typical case of ***cross-attention***, the attended context sequence is separate from the generated sequence for which the attention is used, e.g. when a decoder attends to the output of an encoder. This is accomplished by setting the query to the decoder state $\boldsymbol{q} = \boldsymbol{h}_i$ and $\boldsymbol{K} = \boldsymbol{V}$ to the attended sequence.

In a deep architecture, it is also possible to use as context the sequence itself, or more precisely its state in a previous layer. In ***self-attention*** (Vaswani et al., 2017), the output of the previous layer is used as queries, keys, and values $\boldsymbol{Q} = \boldsymbol{K} = \boldsymbol{V}$. As the whole layer is evaluated in parallel, all queries are concatenated into a single matrix

$$\boldsymbol{Q} = [\boldsymbol{q}_1,\ldots,\boldsymbol{q}_I]. \tag{3.9}$$

When self-attention is used in the encoder, access to future timesteps is not only possible but beneficial. Decoding proceeds autoregressively from left to right, i.e. the output of the previous timestep is fed back in as input to the next timestep. Masking is applied during training to prevent the decoder self-attention from trying to access future information.

A convolutional network applies different parameters depending on the local order, while a recurrent network can use the recurrent state to count the distance from the ends of the sentence, and to account for the local ordering of words. The attention mechanism instead computes the attention distribution for each timestep in isolation, and then summarizes the entire sequence by weighted average. There is nothing in the attention mechanism that would allow it to account for word order. ***Positional encoding*** is applied to make word order information available to the self-attention. A pattern of sine and cosine waves are added to the input embeddings

$$\text{PosEnc}(t,2i) = \sin\left(\frac{t}{10000^{2i/d}}\right), \quad \text{PosEnc}(t,2i+1) = \cos\left(\frac{t}{10000^{2i/d}}\right) \tag{3.10}$$

[6] More on encoder-decoder architectures in Section 5.2.3.

where t is the timestep, $i \in \{0,\ldots,(d-1)/2\}$ defines the dimension, and d is the size of the embeddings.

A single attention mechanism can only effectively attend to a single position in the context sequence. While it is possible for the bulk of the attention distribution to be spread out over multiple timesteps, combining the values at those timesteps through elementwise averaging limits the effectiveness of such divided attention. In practice the attention distributions tend to become peaked (Bahdanau et al., 2014).

Multi-head attention enables attending to multiple timesteps simultaneously by running multiple attention mechanisms in parallel. The queries, keys, and values are projected separately for each attention head. The weighted averaging is also separate for each attention head. The results of the heads are finally concatenated together and projected to the desired dimensionality.

$$\begin{aligned} a_i &= \text{Attention}(\boldsymbol{q}W_i^{\boldsymbol{q}}, \boldsymbol{K}W_i^{\boldsymbol{K}}, \boldsymbol{V}W_i^{\boldsymbol{V}}) \\ \text{MultiHead}(\boldsymbol{q},\boldsymbol{K},\boldsymbol{V}) &= [a_1;\ldots;a_h]W^O \end{aligned} \tag{3.11}$$

To summarize, many approaches for modeling sequential data using neural networks have been presented. The approaches make different trade-offs, and different architectures may be suited for different tasks. However, the self-attention based Transformer architecture (Vaswani et al., 2017) has recently become very popular, particularly for those NLP tasks in which data is abundant.

3.2.3 Discrete and continuous representations

In rule-based and statistical NLP methods, basic units are typically represented using discrete, symbolic representations, meaning that there is a discrete symbol identifying each item. Even if the symbols are discrete, they can have real numbered statistics associated with them, e.g. probabilities based on counting occurrences of a particular pattern in a corpus. Examples include word counts and n-grams, i.e. counts of sequences of n particular words.

When symbols are encoded into a vector using one-hot coding, where each dimension of the vector space corresponds to a single item, the result is a sparse vector representation. To encode an item, only the corresponding dimension is set to one, while all others remain zero. E.g. $[0,0,1,0,\ldots,0]$ could represent the word *"an"*, if it is the third word in the vocabulary. A bag of words can be represented as a "many-hot" coding by combining the individual representations using Boolean OR.

Discrete representations can be called sparse in another sense, as they suffer from ***sparsity of statistics***, when each distinguishable item has its own independent statistics. This means that the representation of a rare item cannot benefit from statistics collected for similar or related items, leading to unreliable representations. For words, if even one character differs, the word is counted as distinct. Even though there might in fact be substantial similarity between the items, no information is shared.

If an item has never been encountered, it will have zero probability, unless a smoothing method is applied to shift probability mass from frequent items to rare and unseen items based on, e.g., lower order distributions. The unsmoothed model is overconfident about its predictions, in particular of the impossibility of unseen events. Smoothing can be guided by knowledge-based ***feature engineering***, by generalization based on class labels or clustering. E.g. if a particular pattern *"I talked with"* is commonly followed by a proper name, the probability of all proper names could be increased even for names not observed with that particular pattern. As a data-driven alternative, the standard smoothing method for n-gram language models is modified Kneser-Ney smoothing (Kneser and Ney, 1995). Smoothing can only partly alleviate the problem of sparsity.

A more robust solution is found by exploiting the distributional hypothesis, defining the meaning of an item through the contexts in which it occurs. The item is represented as a vector in a dense high-dimensional space. Similarities between items can be computed as distances in this vector space. Directions in the vector space correspond to compositional aspects of meaning, preferably in such a way that adding a particular vector offset causes the same change in meaning everywhere in the space. The meaningful directions span lower-dimensional manifolds embedded in the high-dimensional space, on which the words are arranged. This is perhaps most evident when physical continua, e.g. color, temperature, and size, are discretized into words. The properties of meaningful directions and offsets allows constructing new examples using vector arithmetic. While the constructed vector does not exactly match any observed word, the nearest neighbor(s) to the constructed point can be retrieved. Arithmetical vector semantics is essential for natural language generation, as it enables constructing a vector representation as a function of the context, and then predicting a vocabulary item conditioned on the constructed vector.

End-to-end learning with continuous representations automates feature generation, eliminating the need for manual feature engineering which has been a central task in ML engineering. The embeddings are trainable parameters. A central idea in deep learning is the principle of multiple levels of composition. Deep architectures are able to learn hierarchical representations, in which the first layers of a network learn general low-level features (e.g. edges, corners), and deeper layers compose those features into more specialized high-level patterns (e.g. eyes, wheels), which in turn are composed to patterns of an even higher level (e.g. faces, cars). This specialization is learned automatically, without the need for a human to determine what each layer should be learning.

Discrete representations cause a bias towards short context windows and large basic units, even at the cost vocabulary growth, as the state space grows exponentially w.r.t. sequence length, but only linearly w.r.t. vocabulary size. Due to Zipf's law, large vocabularies cause more items to be rare and therefore poorly modeled. Deep neural networks make powerful use of continuous representations,

making them more robust towards sparsity.[7] Continuous representations do not suffer from the previously mentioned bias, but instead prefer small vocabularies of frequent items.

3.3 Loss functions

Machine learning involves solving a model selection problem: from a set of alternatives the machine learning practitioner needs to select a particularly good alternative. In parametric learning the alternatives are values for the model parameters.

The ***loss function*** determines how good particular values for the parameters are. The optimal parameters are a global minimum of the loss function. It is also possible for the loss function to have local minima: points where the loss is smaller than for any point within some neighborhood, but larger than the global minimum. Local minima may be acceptable if they are close in value to the global optimum. If not, a large search error occurs.

If the optimization problem admits a closed-form solution, the optimal parameters can be found in a single step. Often there is no known closed-form solution, making it necessary to use an iterative learning procedure instead. The iterative procedure can be based on heuristics or numerical optimization. This optimization can be described as a ***search*** for the optimal parameters. During the search, new candidate parameters are generated e.g. from the previous best parameters, or from a model of the loss surface. The candidate parameters are evaluated using the loss function.

If the loss function is convex, it has no local minima that are not the global minimum, and the search is guaranteed to find the global optimum. If local minima are present, it is possible for the search to get stuck in them.

3.3.1 Maximum Likelihood Estimation

A point estimate for the parameters is a finite dimensional vector $\boldsymbol{\theta} \in \Theta$ from a parameter space Θ. The likelihood is the probability of the data given the parameters $P(\boldsymbol{D} \mid \boldsymbol{\theta})$. As the likelihood is used during training, the data is fixed but the optimal parameters are not yet known. Thus the likelihood is a function of the parameters.

Selecting the optimal parameters as a point estimate maximizing the likelihood function is called ***maximum likelihood*** estimation

$$\hat{\boldsymbol{\theta}}_{ML} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{D} \mid \boldsymbol{\theta}). \tag{3.12}$$

Usually the full data likelihood is the product of likelihoods of independent and identically distributed (iid) samples, in which case we optimize the logarithm of

[7] While continuous representations can be used in non-neural methods, it is less common.

the likelihood for computational reasons. The logarithm turns the product into a sum, and conserves numerical precision by avoiding very small numbers in the results of the computations.

3.3.2 Maximum a Posteriori Estimation

We may have some prior knowledge of where we expect the optimal parameters to be found. This prior knowledge can be expressed formally by giving a probability distribution over the parameters. Bayes' theorem describes how to update a ***prior*** belief, represented as a probability distribution $P(A)$ based on some new observation B

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}. \tag{3.13}$$

The updated belief is called the ***posterior*** distribution. In maximum a posteriori (MAP) estimation, the value maximizing the posterior distribution is selected as a point estimate for the parameters

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta} \mid \boldsymbol{D}) = \arg\max_{\boldsymbol{\theta}} \frac{P(\boldsymbol{D} \mid \boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\boldsymbol{D})}. \tag{3.14}$$

Even though MAP estimation uses Bayes' theorem, it differs from a full Bayesian approach. The use of a point estimate results in discarding information about the uncertainty of the estimate. A full Bayesian approach foregoes the use of point estimates, using the full posterior distribution during the entire inference. The use of a point estimate may be motivated by the need to limit computational cost.

3.3.3 Minimum description length

The minimum description length (MDL) principle (Rissanen, 1978, 1989) is core idea behind several approaches for constructing loss functions. It is based on the observation that both learning and compression depend on exploiting regularities in the data. In compression, the regularities are a means to an end, as the goal is the maximally compressed data. In MDL-style machine learning, the model expressing the regularities is the goal, and the compression is merely instrumental.

In ***ideal MDL***, the code is in the form of a universal (e.g. Turing) machine which when run yields the data. The length of the shortest such machine is the ***algorithmic complexity*** of the data. Unfortunately this complexity is not computable. ***Two-part MDL*** (Rissanen, 1978) is a practical variant. The goal is to encode a particular data set $\boldsymbol{D}$, assumed to be generated from a distribution $P(\boldsymbol{D})$. The encoding should require the minimum possible number of bits, while still being losslessly decodable. The encoding is based on a hypothesis, here represented by the parameters $\boldsymbol{\theta} \in \Theta$. The hypothesis space Θ and the conditional probability distribution $P(\boldsymbol{D} \mid \boldsymbol{\theta})$ are assumed to be known by both encoder and decoder, but the selected point hypothesis $\boldsymbol{\theta}$ is not. To communicate the point hypothesis to the decoder, it must first be encoded using a description of length

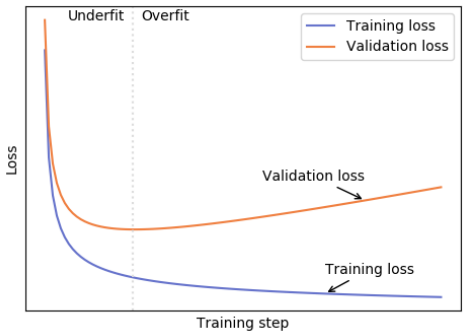


Figure 3.3. Schematic illustration of learning curves. Training loss keeps decreasing, while validation loss starts increasing when overfitting begins.

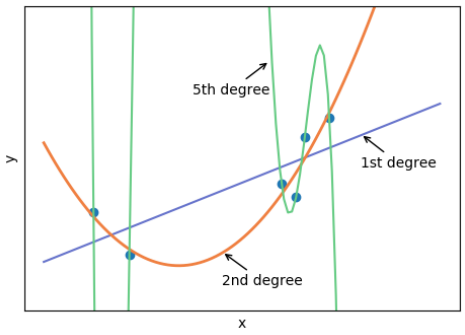


Figure 3.4. Underfitting (1st degree polynomial) and overfitting (5th degree polynomial) in 1-d regression to noisy data generated from a 2nd degree polynomial.

$\mathbf{L}(\boldsymbol{\theta})$ to form the first part of the code. The second part is the description of the data when given the point hypothesis $\mathbf{L}(\boldsymbol{D} \mid \boldsymbol{\theta})$. The best point hypothesis minimizes the sum of the lengths of the two parts

$$\mathbf{L}(\boldsymbol{D}, \boldsymbol{\theta}) = \mathbf{L}(\boldsymbol{\theta}) + \mathbf{L}(\boldsymbol{D} \mid \boldsymbol{\theta}). \tag{3.15}$$

Shannon's source coding theorem (Shannon, 1948) states that the optimal coding for $\mathbf{L}(\boldsymbol{D} \mid \boldsymbol{\theta})$ has length $-\log_2(\boldsymbol{D} \mid \boldsymbol{\theta})$. While MDL leaves room for choice in how the hypothesis is coded, if a complete code with length $\mathbf{L}(\boldsymbol{\theta}) = -\log_2(\boldsymbol{\theta})$ is used, two-part MDL becomes equivalent to the MAP-estimate with prior $\boldsymbol{P}(\boldsymbol{\theta})$.

3.3.4 Regularization

The machine learning practitioner is faced with difficult questions: How do we know that we are done training? Did we make good decisions when specifying the capacity of the model? How much should we trust our data? A good model should generalize robust patterns in the data, but not bend out of shape to fit noise and outliers.

A simple way to measure generalization is comparing the curves of the loss function on training and held-out validation data sets, as shown in Figure 3.3. In the ***underfitting*** regime, both losses are higher than optimal. Perhaps training is not yet complete, in which case the losses should be decreasing over time. If underfitting occurs when converged, the model may be too rigid, and thus unable to approximate the desired distribution well. In the opposite ***overfitting*** regime, training loss becomes very small or even zero, but validation loss is high. The model is too flexible, allowing it to overfit the training data, which hurts generalization to unseen data points.

The capacity of the model can be controlled by adjusting the amount of parameters to achieve a good fit. Figure 3.4 demonstrates this using polynomial regression. Another way is to use ***regularization***, in which the loss function is

modified in a way that encodes some prior knowledge of what a good fit looks like. The form of the regularization depends on what the prior knowledge is.

The intuition that weights should not be too large leads to L_p ***regularization***, in which the L_p norm of the weight matrix is added to the loss. The L_2 norm is the Euclidean distance. The L_1 norm, or Manhattan distance, also causes sparse weights to be learned. L_p regularization is a special case of Tikhonov regularization (Tikhonov, 1963), which can also express e.g. certain smoothness constraints.

Another intuition is that the model should be robust to small perturbations of the input or intermediary representations. The use of ***noise as regularization*** is a successful technique used e.g. in Dropout (Srivastava et al., 2014) and SwitchOut (Wang et al., 2018).[8] In Dropout, some dimensions of a neural encoding vector (an embedding, or a hidden state vector) are stochastically zeroed out during training. This encourages learning of robust, redundant features, instead of relying on the presence of a single informative feature. ***Subword regularization***, a noise based technique proposed by Kudo (2018), is discussed in Section 5.5.11.

The model should not be overconfident in its predictions. This intuition leads to ***label smoothing*** (Szegedy et al., 2016). In standard maximum likelihood estimation of a categorical distribution with a gold label, the loss is the cross-entropy with a Dirac δ function, i.e. the best prediction assigns all the probability mass to the correct label. Smoothing the δ target distribution by distributing some of the probability uniformly to all incorrect labels penalizes overconfidence. The δ target distribution is particularly ill-suited for use with a softmax predictor, as the softmax can only approach, but never achieve, the desired distribution. This means that even if the distribution is already a good fit, each training example will move the weights to even more extreme values, which will ultimately lead to numeric overflow. In addition to preventing overfitting and improving numerical stability, label smoothing also helps in retaining the flexibility of the model to adapt to new data, which is difficult to do if the weights become extreme.

Even ***early stopping*** of training acts as regularization. When using an iterative optimizer together with an overparameterized model, training progresses over time from underfitting to overfitting, as seen in Figure 3.3. Stopping training at the correct step thus prevents overfitting.

Also ***multi-task learning*** setups[9] have a regularizing function by claiming some of the capacity of the model. The model is prevented from overfitting to small data for a low-resource task, as the shared parameters must also fit the high-resource task(s) well.

[8] SwitchOut discussed in Section 5.5.11

[9] See Section 3.4.2 for multi-task learning.

3.4 Learning setups

In machine learning, we may have access to various types of data with or without annotations. Let us divide the training data $\boldsymbol{D}$ into features $\boldsymbol{X}$ and labels $\boldsymbol{Y}$. During testing or production use, the features will be the inputs to the system, and the labels are the output.

Machine learning setups differ in the structure of $\boldsymbol{X}$ and $\boldsymbol{Y}$, and in how the training labels are available to the learning algorithm.

Supervised learning is the most common learning setup. In supervised learning, the entire training data $\boldsymbol{D}$ is provided with labels $\boldsymbol{Y}$. The abundance of labels makes it straight-forward to train models discriminatively to predict the labels given the features. Supervised training is limited to tasks in which enough labeled data is available. When sufficient labels are expensive to collect, supervised learning becomes infeasible.

Unsupervised learning describes a setup where no labels are provided. A model for $\boldsymbol{X}$ is constructed, with $\boldsymbol{Y}$ as a latent variable. Generative models are particularly suited to unsupervised learning, when it is not known in advance what patterns should be found in the data.

The term ***self-supervised learning*** has recently gained popularity (e.g. Lan et al., 2019; Liu et al., 2019; Joshi et al., 2020). While there is no clear boundary between unsupervised and self-supervised learning, the latter emphasizes the use of richly informative but naturally occurring data. The unlabeled data is partitioned into two disjoint subsets[10], which are used as features and labels respectively. In NLP, this can e.g. describe a masked language model loss, in which the surrounding sentence context is used to predict a masked item.

Semi-supervised learning is an intermediate setting between unsupervised and fully supervised learning, in which only a part of the training data is labeled

$$\boldsymbol{D} = [(\boldsymbol{X}^{(L)}, \boldsymbol{Y}^{(L)}); (\boldsymbol{X}^{(U)}, \emptyset)]. \tag{3.16}$$

When the proportion of labeled data is very small, the learning setup is sometimes called ***weakly supervised learning*** or minimally supervised learning.

3.4.1 Active learning

In normal semi-supervised learning, all available labels are provided immediately at the start of training. In active learning (AL) the labels are made available incrementally. A ***labeling oracle*** provides correct labels for individual examples. The ***labeling budget*** determines how many times the oracle may be invoked.

Active learning can be particularly useful for collecting annotations. Data annotation is often performed for the specific goal of improving the performance of a particular system on a task. This gives the opportunity to carefully select the

[10] A new partition can be made for each parameter update.

data that will be annotated, in order to maximize the effect and minimize the cost of collecting the annotations.

Active learning methods can be divided into three frameworks: pool-based active learning, stream-based selective sampling and (membership) query synthesis (Settles, 2010).

In ***pool-based active learning*** (Lewis and Gale, 1994), the system can select samples to be labeled from an unordered pool of unlabeled data $\mathscr{A}$. Samples can be selected either one at a time (on-line) or in a larger batch, before updating the information available to the learner. Pool-based active learning is applied in Publication III.

In ***stream-based selective sampling***, the learner can not select samples freely. Instead the potential samples are shown one by one, and the learner has to decide for each sample whether to elicit an annotation or not.

The third type of active learning is called ***query synthesis***. In query synthesis, there is no pool of candidates. Instead labels are elicited for samples drawn from the generative model, which must be good enough to create realistic samples.

The pool-based active learning procedure is defined as follows: In each iteration, a query strategy selects the next samples to elicit

$$\boldsymbol{X}_{t+1} = \textsc{Strategy}(\mathscr{A}, \boldsymbol{D}^{(U)}, \boldsymbol{D}^{(L)}_{1:t}, \mathrm{M}_t). \tag{3.17}$$

The query strategy has access to four sources of information:

1. the training pool $\mathscr{A}$,
2. the set of unannotated data $\boldsymbol{D}^{(U)}$,
3. the current set of annotated data $\boldsymbol{D}^{(L)}_{1:t}$,
4. and the current best model M_t, trained with $\boldsymbol{D}^{(U)}$ and $\boldsymbol{D}^{(L)}_{1:t}$.

The labeling oracle provides the true label(s) $\boldsymbol{Y}_{t+1}$ for the selected samples, which are added to the labeled training data. A new iteration of the model is trained. The selection strategy can make use of the knowledge learned by the current iteration of the model.

Uncertainty sampling (Lewis and Gale, 1994) is the most prominent query strategy. It uses the model's estimate of the uncertainty of the prediction for each sample as the selection score. The intuition is that the label for samples of which the model is uncertain are likely to be more informative than samples of which the model is certain. As a downside, noise and outliers are likely to have uncertain predictions while being uninformative, which may affect the robustness of the method.

While uncertainty sampling compares the probability of the selected label to the sum probability of all other labels

$$\boldsymbol{X}_{t+1} = \operatorname*{arg\,max}_{\boldsymbol{X}} \big(1 - P(\hat{\boldsymbol{Y}}_1 \,|\, \boldsymbol{X})\big), \tag{3.18}$$

the related margin sampling only compares it to the runner up

$$\boldsymbol{X}_{t+1} = \operatorname*{arg\,min}_{\boldsymbol{X}} \big(P(\hat{\boldsymbol{Y}}_1 \,|\, \boldsymbol{X}) - P(\hat{\boldsymbol{Y}}_2 \,|\, \boldsymbol{X})\big). \tag{3.19}$$

Entropy-based uncertainty sampling uses the entire distribution over labels

$$\boldsymbol{X}_{t+1} = \arg\max_{\boldsymbol{X}} \left(-\sum_i P(\boldsymbol{Y}_i \,|\, \boldsymbol{X}) \log P(\boldsymbol{Y}_i \,|\, \boldsymbol{X}) \right). \tag{3.20}$$

Density-weighted methods apply density estimation in order to avoid selecting outliers. Combined with uncertainty, they lead to selecting samples which are informative in two ways: in addition to the model being uncertain, the sample comes from a dense, representative region of the feature space. One density-based method, ***representative sampling*** (Xu et al., 2003), selects samples which are dissimilar to each other, in order to give a good coverage of the dataset. The samples are clustered using k-medoids, and the cluster medoids are returned as the selected samples. Selecting dissimilar samples is of particular importance when selection and training is done in batches instead of on-line. An on-line algorithm updates the uncertainty after each sample, making it less likely to select redundant words than a batch algorithm.

For more about active learning, see Settles (2010) or Guyon et al. (2011).

3.4.2 Transfer and Multi-task learning

The availability of multiple related tasks opens up the possibility of ***transfer***, when knowledge gained while learning one task is transferred to another. The tasks can either be trained sequentially, in parallel, or something in between. For historical reasons, sequential transfer is sometimes called just ***transfer learning*** (Pratt et al., 1991), while parallel transfer is called ***multi-task learning*** (MTL) (Caruana, 1998). Following Arivazhagan et al. (2019b) and Dabre et al. (2020), I use transfer learning as a wider term subsuming both types of transfer. For a survey on transfer learning for classification, regression, and clustering problems, see Pan and Yang (2010).

In this training setup, the training data is the combination of $\boldsymbol{N}$ tasks $[(\boldsymbol{X}^{(1)},\boldsymbol{Y}^{(1)}),\ldots,(\boldsymbol{X}^{(N)},\boldsymbol{Y}^{(N)})]$. Typically one task is the primary task, and the other tasks are supporting auxiliary tasks. In sequential transfer, the term parent and child are used for the auxiliary and primary tasks, respectively. Transfer can be particularly useful in situations where limited training data is available for the primary task, but data for the auxiliary tasks is abundant.

Multilingual training can be seen as a multi-task setting in which each language pair in the training data is a separate learning task (Luong et al., 2015a). In this domain the setting can be called ***cross-lingual learning***. Typically some information is transferred from a high-resource to a low-resource language.

Cross-lingual learning differs from ***annotation projection*** in the type of the end result. In annotation projection, the outcome of the procedure is a set of annotations, e.g. for Part-of-Speech tagging or parsing. The annotations are mapped from a high-resource to a low-resource language. In cross-lingual learning, the outcome is instead a model capable of performing some task.

Torrey and Shavlik (2009) describe three ways in which transfer learning can benefit training: 1) higher performance at the very beginning of learning, 2)

steeper learning curve, and 3) higher asymptotic performance. The first two are of importance when rapid training is essential, while the third is critical in the common case where computational constraints are less pressing.

Learning multiple tasks can be seen as a form of regularization.[11] There may be non-robust features that are highly predictive in a small training set, but do not generalize. The auxiliary tasks help in preventing the model from overfitting to these incidental features for the primary task. The model must instead learn representations that generalize well to the combination of tasks. A possible goal is that the joint representations have a higher level of abstraction. ***Interlingual*** representations are shared between multiple languages.

Not all transfer is beneficial. Negative transfer, also called ***interference***, occurs when the representations needed to solve different tasks are too different, and the model capacity is insufficient to represent both. There is research into the effects of negative transfer and the importance of the choice of auxiliary tasks (Rosenstein et al., 2005; Bingel and Søgaard, 2017; Bjerva, 2017; Deleu and Bengio, 2018).

Sequential transfer is a form of adaptation. In sequential transfer learning, the ***pretraining*** on a high-resource parent task is used to initialize and constrain the ***fine-tuning*** training on the low-resource child task. Zoph et al. (2016) apply sequential transfer learning to low-resource neural machine translation. There are various ways in which knowledge can extracted from the parent model. Typically all or part of the parameters are transferred into the child model, e.g. by initializing embeddings of an NLP system using a model pretrained on a different task.

In ***parallel transfer***, tasks are trained simultaneously or their training is interleaved. In simultaneous training, a single parameter update contains information from multiple tasks. This is particularly useful when a single input is labeled for several tasks, and part of the computation can be reused. It is also common to build mixed minibatches consisting of examples from multiple tasks. If the computations required for different tasks differ too much, it may be preferable to interleave training by alternating between the tasks.

Sequential transfer carries the risk of ***catastrophic forgetting*** (McCloskey and Cohen, 1989; Goodfellow et al., 2014), in which the knowledge gained from the first task fades away completely. If it is known which parameters extract the generic features, overfitting may be delayed by freezing those parameters between the two training phases. This reduces the number of parameters trained from the low-resource data. With parallel transfer, catastrophic forgetting does not occur.

In sequential transfer, the intended task can only start affecting the representations once pretraining is finished. Parallel transfer may see a benefit for learning general representations, as it can learn from all tasks throughout training. However, parallel transfer requires the task mixture weights to be tuned, so that learning progresses in the correct pace. Finding a good balance may be challeng-

[11] More on regularization in Section 3.3.4.

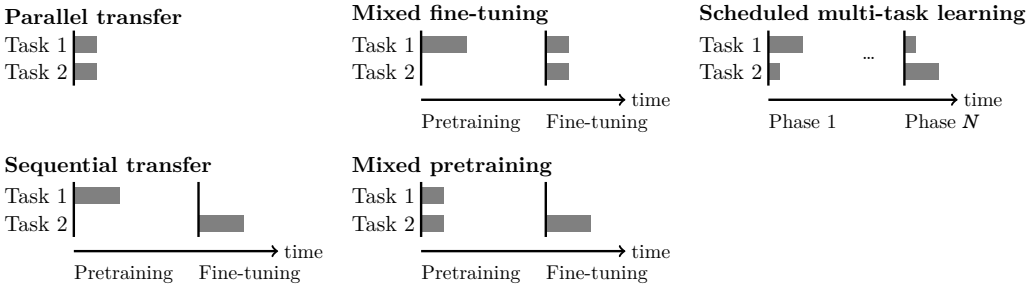


Figure 3.5. Task mixing strategies for transfer learning.

ing when tasks have very different level of difficulty, or the amount of data for different tasks is highly asymmetrical. Sequential transfer does not require the same tuning, as convergence can be determined for each task separately.

Sequential transfer is preferable in situations where the final task is not known at pretraining time, or the training data for it is not yet available. One example is developing systems that can be rapidly adapted to a newly interesting low-resource language for use during a crisis such as a natural catastrophe. In this scenario there may not be time to start the costly parallel training procedure from the beginning. One consequence of such sequential transfer is that preprocessing, in particular the subword segmentation, cannot be optimized for the low-resource language, as it is not known at pretraining time.

It is also possible to combine sequential and parallel transfer. Figure 3.5 shows some possible ways of achieving this by mixing the tasks. One strategy, ***mixed fine-tuning***, begins with pretraining only on the high-resource task, and continues fine-tuning with a mixture of tasks. Chu et al. (2017) apply this strategy to domain adaptation. The inverse setting, called ***mixed pretraining***, is used by Kocmi (2019). Pretraining is performed on a mixture of tasks, while fine-tuning only uses the child task.

Kiperwasser and Ballesteros (2018) propose generalizing these strategies into ***scheduled multi-task learning***, in which training examples from different tasks are selected according to a mixing distribution that changes during training according to the task-mix schedule. They experiment with three schedules: constant, exponential, and sigmoidal. In Publication IX, a new partwise constant task-mix schedule suitable for an asymmetric-resource setting with multiple auxiliary tasks is proposed. The task-mix schedule can have an arbitrary number of steps, any of which can be mixing multiple tasks. All the other strategies can be recovered by using particular schedules with scheduled multi-task learning. Use of a fine-grained scheduled multi-task learning requires the task mixing to occur at training time, which precludes use of a preprocessing step for the oversampling and task mixing.

Scheduled multi-task learning also enables a training scheme that could be used in the rapid adaptation scenario: a multilingual parent model is pretrained in advance on multiple high-resource tasks. When the need arises, the model can

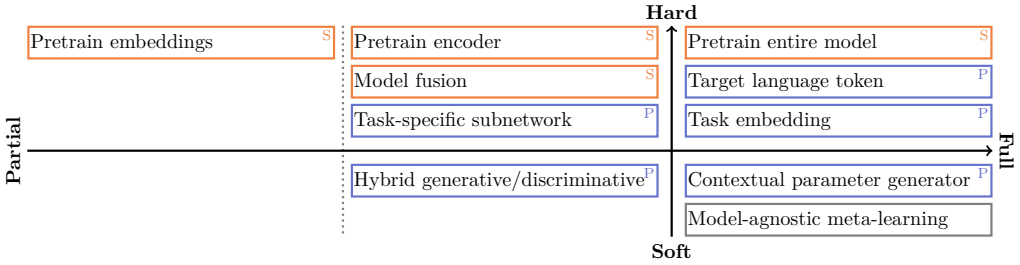


Figure 3.6. Methods for transfer, arranged by type of parameter sharing. Orange and blue borders indicate sequential (S) and parallel transfer (P), respectively.

be sequentially transferred to a new low-resource task.

Transfer is essential in ***asymmetric-resource*** settings, in which the amount of training examples for the target task very small, requiring the learner to rapidly generalize. Humans are very adept at this, often being able to generalize correctly even from only a single training example. Based on the number of training examples for each new target task, the learning setups are also called ***few-shot***, ***one-shot*** and ***zero-shot*** learning. Note that background data is needed in addition to the training examples for the target task. In these learning setups, the background data is labeled for other tasks, while in the related setup of weakly supervised learning, the background data is unlabeled. For a survey on few-shot learning, see Wang et al. (2020).

The aim of ***meta-learning***, also called learning-to-learn, is to generalize from background training data in order to quickly adapt to new tasks. In a meta-learning system there is typically some division into slow and fast learners with distinct parameters. The slowly changing meta-learner learns to quickly adapt the fast learner to new tasks. The term ***life-long learning*** (Thrun, 1995) describes a scenario in which there is no separation between training and testing phases, and the learner is expected to keep learning from new experiences indefinitely.

In ***knowledge distillation***, a student model is trained to approximate the output distribution of a teacher model. Typically the aim is model compression, and the student is a smaller network trained on the same data. Chen et al. (2017) apply knowledge distillation to machine translation by using as parent another MT system trained to translate into the same target language from a different pivot language.

Methods for transfer learning in NLP

Transfer is typically implemented through ***parameter sharing***. In sequential transfer, the parameters trained on the parent task are used to initialize those of the child task, while in parallel transfer the dependency between parameters is in the form of a constraint. It is possible to share all parameters, or select a subset for sharing, allowing the remaining ones to be task-specific. For example pretraining the entire model on a related task is ***full parameter sharing***, while pretraining only the embeddings or a subnetwork, e.g. the encoder, is ***partial pa-***

rameter sharing. As embedding or encoder pretraining is typically performed on a generic contextual prediction task that differs from the target task, this is a form of sequential transfer. Parameter sharing can be controlled on a fine-grained level (Sachan and Neubig, 2018). Shared attention (Firat et al., 2016a) uses language-specific encoders and decoders with a shared attention, while language-specific attention (Blackwood et al., 2018) does the opposite by sharing only the feedforward sublayers of the decoder, while using language-specific parameters for the attention mechanisms. ***Model fusion*** combines the predictions of an entire pretrained model for an auxiliary task, e.g. a language model, into the predictions of the model being trained.

Parameter sharing can be either hard or soft (Ruder, 2017). In ***hard parameter sharing*** the exact same parameter matrix is used for several tasks. In ***soft parameter sharing***, also called parameter tying, each task has its own parameter matrix, but a dependence is constructed between the corresponding parameters for different tasks. It is e.g. possible to regularize the parameters to be close by adding a norm penalty to the loss function. Hard parameter sharing benefits from the shared parameters being exactly equal, which means that only one copy needs to be stored in memory. This can result in a large reduction in the memory footprint of the model. Lasserre et al. (2006) use soft parameter sharing between the priors of a generative and discriminative model to enable a principled way to interpolate between the two types of training.

The ***target language token*** (Johnson et al., 2017) and ***language embedding*** (Conneau and Lample, 2019a) approaches use hard sharing of all parameters. In the former, the model architecture is the same as in a language-pair-specific model. The target language is indicated by a special target language token, prepended to the input by a preprocessing step. E.g. adding ⟨TO_FI⟩ indicates that the target language is Finnish. The approach can be scaled to more languages by increasing the capacity of the model, primarily by increasing the depth in layers (Arivazhagan et al., 2019b). The language embedding can be described as a factored representation, with the extra factor marking the language of each word on the target side.

Model-agnostic meta-learning (MAML) (Finn et al., 2017) meta-learns a set of initial parameters that only requires a small number of gradient steps to adapt to a new task. Vinyals et al. (2016) apply meta-learning to language modeling, Kaiser et al. (2017) to machine translation. Gu et al. (2018b) apply meta-learning to find initializations that can very rapidly adapt to a new low-resource source language.

Learning to share is a way to accomplish meta-learning using soft parameter sharing. The ***contextual parameter generator*** (Platanios et al., 2018) consists of two neural networks: the model and the parameter generator. The parameter generator generates the weights of the model from some contextual variables. As the model parameters are generated values and not trainable parameters, the gradient of the loss must be backpropagated through both networks. The parameter generator meta-learns a soft dependency between parameters for different tasks.

Kang et al. (2011) identify task groups for parameter sharing using mixed integer programming. Zaremoodi et al. (2018) use a trainable routing network to enable adaptive sharing of subnetworks.

Figure 3.6 categorizes some methods for transfer learning based on the hardness and extent of parameter sharing and whether the methods are sequential or parallel.

3.5 Learning algorithms

The formal expression of a machine learning task often involves two optimization operations. During training, the optimal parameters are found with a ***search*** over the space of parameters

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Theta} \mathrm{L}(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{\theta}). \tag{3.21}$$

At test time, the output is decoded using a search over the hypothesis space

$$\hat{\boldsymbol{Y}} = \arg\max_{\boldsymbol{Y}\in\mathscr{Y}} P(\boldsymbol{Y} \mid \boldsymbol{X}; \boldsymbol{\theta}^*). \tag{3.22}$$

The search spaces, Θ and $\mathscr{Y}$, from which the optimal value needs to be found, are typically very large. It is not feasible to perform exhaustive search iterating over all the possible values. Instead a search algorithm must be applied.

Depending on both the loss surface and the search, this can either result in a global optimal value or some sort of approximation, e.g. a locally optimal value. When choosing the search algorithm, one must consider both computational and memory complexity, the difficulty of the loss function, the acceptableness of deviations from the true optimum, and whether there exists a good heuristic for estimating the true cost of making a decision.

In ***local search***, the search proceeds through iterative refinement of the current best hypothesis. The neighborhood of the current hypothesis is explored by taking small search steps, from which one is chosen as the next hypothesis. This can be contrasted against ***global search***, in which the next hypothesis is chosen from anywhere in the search space, using a global estimate of the loss function.

In ***greedy search***, the local search step with the lowest loss is always taken, and any history of prior hypotheses is discarded. The aggressive pruning keeps the number of considered successor states small, and thus the search quickly progresses towards a goal state. As a downside, greedy search can get stuck in "dead ends", or paths that begin with low-loss steps, but continue with only high-loss alternatives. Greedy search is unable to back out of the "dead end", resulting in reaching a suboptimal goal.

An ***optimal search*** is guaranteed to prune hypotheses only once they have been proven suboptimal. When a "dead end" is encountered, the search can backtrack to an earlier hypothesis and make a different choice. As a downside, the data structure containing the information for backtracking can require large amounts of memory.

Beam search is a compromise between optimal and greedy search. Multiple hypotheses are kept, while bounding the memory requirements. At each timestep, all current hypotheses are expanded with their possible continuation steps. The list of hypotheses is sorted in increasing order of the scoring function, usually the loss. The hypotheses are then pruned, keeping only a fixed number (called the ***beam width***) of best hypotheses. Any hypotheses with scores worse than the ones in the beam are pruned, meaning that the search can no longer backtrack to them. As it is possible that the optimal path falls out of the beam, the search is not optimal.

If an infinite beam width is used, the optimal ***best-first search*** is recovered. Setting the beam width to one results in greedy search.

The ***heuristic*** is an estimate of the loss of completing a hypothesis. This loss can be divided into two parts

$$\mathrm{L}(\hat{\boldsymbol{Y}}) = \mathrm{L}(\boldsymbol{Y}_{0:t}) + \mathrm{L}(\boldsymbol{Y}^*_{t+1:T}) \tag{3.23}$$

where the first part is the known loss of the already predicted labels up to the current timestep t, and the second part is the heuristic future loss of the optimal labels from the next timestep $t+1$ to the end T. Beam search ***heuristic penalties*** are typically not admissible heuristics, but rather try to penalize hypotheses that seem at risk of becoming "dead ends".

3.5.1 Expectation-Maximization

The Expectation Maximization (EM) algorithm (Dempster et al., 1977) is an algorithm for finding ML estimates for parameters in graphical models with latent variables, e.g. mixture models. The training data for such a model is only partly observed, with the values for the latent variables missing. An iteration of the EM-algorithm is guaranteed not to decrease the likelihood of the data, which leads to guaranteed convergence (Dempster et al., 1977). However, convergence may be slow, and the algorithm may converge to a poor local optimum if the loss surface is difficult or the initial values are unfortunate.

The EM algorithm is an iterative algorithm alternating between two steps. The aim is to maximize the complete data likelihood, including both the observed $\boldsymbol{D}$ and the latent variables $\boldsymbol{Y}$. As the value for $\boldsymbol{Y}$ is not observed, we must take the expected value of the complete data likelihood under the current posterior of the latent variables. This is called the E-step,

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}_{i-1}) = \int_{\boldsymbol{Y}} \log P(\boldsymbol{D}, \boldsymbol{Y} \mid \boldsymbol{\theta}) P(\boldsymbol{Y} \mid \boldsymbol{D}, \boldsymbol{\theta}_{i-1}) d\boldsymbol{Y} \tag{3.24}$$

In the M-step, the parameters are updated to maximize the expected value computed in the E-step:

$$\boldsymbol{\theta}_i = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}_{i-1}). \tag{3.25}$$

With the modification of adding the prior to the maximization in the M-step, EM can also find MAP estimates (Bishop, 2006).

A typical application of EM is clustering with a Gaussian Mixture Model (GMM), which results in a soft assignment of data points to clusters. When applied to a hidden Markov model, EM is called the forward-backward algorithm. Using instead the related Viterbi algorithm (Viterbi, 1967) is sometimes referred to as *hard-EM*. In the clustering task, an analogy can be drawn to using the k-means algorithm, which yields a hard assignment of data points to clusters. Spitkovsky et al. (2011) present lateen-EM, a hybrid variant in which EM and Viterbi optimization are alternated. Alternating between soft and hard assignments has the potential of avoiding local optima or areas of slow convergence. When EM is used for clustering, the initial values are typically found using k-means, which can be seen as analogous to a single lateen step.

3.5.2 Stochastic Gradient Descent

Gradient descent (Cauchy, 1847) is a learning algorithm based on local search. The current parameters are improved by taking a small step in the direction in which the loss decreases most rapidly: the direction of the negative gradient of the loss function, evaluated at the current parameters

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \epsilon \nabla \mathrm{L}(\boldsymbol{D}; \boldsymbol{\theta}_t). \tag{3.26}$$

The length of the step is controlled by the learning rate ϵ.

To calculate the true gradient, a pass over the entire training set $\boldsymbol{D}$ is required. If the training data is large, this results in unacceptably large computational cost.

Assuming that the loss decomposes to a sum of losses over the individual examples, a simple solution is to approximate the gradient using a small sample from the data

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \epsilon \nabla \mathrm{L}(\boldsymbol{D}_t; \boldsymbol{\theta}_t); \quad \boldsymbol{D}_t \subset \boldsymbol{D}. \tag{3.27}$$

This is called ***stochastic gradient descent*** (Widrow and Hoff, 1960).

Estimating the gradient from a single data point is also not optimal, as the estimated gradient is very noisy, potentially slowing down convergence. In addition to this, modern GPU hardware is highly effective at performing multiple similar computations in parallel. Computing the gradient from a single datapoint is not much faster than using a small sample, known as a ***minibatch***. Typical minibatch sizes at the time of writing are approximately a few hundred datapoints, limited by the available memory on the GPU. To enable larger effective minibatches, ***gradient accumulation*** can be applied. Gradients from multiple consecutive minibatches are accumulated before applying a parameter update, which is equivalent to using a larger minibatch size.

A global search can attempt to jump directly to the $\boldsymbol{\theta}$ for which $\nabla \mathrm{L}(\boldsymbol{D}; \boldsymbol{\theta}) = 0$ using ***higher-order statistics*** of the loss function. Such higher-order optimization methods have large memory footprints, and are not reliable in the presence of noisy loss function estimates. Noise may be caused both by the stochasticity of SGD and explicitly added as regularization. Gradient descent is limited to

optimization of continuous parameters. It can be generalized to local search in discrete spaces, and is then called ***hill climbing***.

Challenges in gradient-based learning

Gradient descent is sensitive to the learning rate ϵ hyper-parameter. Too large values result in overshooting the optimal path, and even divergence. Too small values cause slow convergence. To complicate the situation even more, the sensitivity of different parameters can vary, causing them to have different optimal learning rates. The ***Adam*** optimizer (Kingma and Ba, 2015) attempts to address the problem by computing individual adaptive learning rates for different parameters from estimates based on a running average of the first and second moments of the gradients. Even though using Adam results in a form of overall step size annealing, it is often combined with a learning rate schedule. A typical choice is linear warmup followed by exponential decay.

The nonlinear functions used in deep learning, e.g. the sigmoid, tanh, rectified linear unit (ReLU) (Nair and Hinton, 2010), and Gaussian error linear unit (GELU) (Hendrycks and Gimpel, 2016), are approximately linear for a certain range of inputs, while ***saturating*** to constant or nearly constant when inputs move far from this range. In the linear region, the unit is sensitive to the input. Linear functions have a useful property: their gradient is constant, making gradient based optimization very stable. In the saturated region the function is (nearly) constant, and the gradient is zero or very small. The input needs to be changed by a large amount in order to cause a noticeable change in the output. This means that it is (nearly) impossible for gradient descent to modify the behavior of the unit. Learning is most effective when units are initialized to start in their linear region, and remain there during the early phases of learning.

In deep neural networks, there are multiple layers with linear weight matrices and nonlinear activation functions between the input and output. The repeated multiplication with a weight matrix makes it very unlikely that the variance of the gradient is retained (Bengio et al., 1994). If the eigenvalues of the weights are larger than 1, the ***gradient explodes*** to very large values. If the eigenvalues of the weights are smaller than 1, the ***gradient vanishes*** to zero. Both prevent effective learning at the earliest layers of the network. Parameter sharing, e.g. of the type used in recurrent neural networks, can exacerbate this problem. In these networks, the same weight matrix is repeatedly applied.

Cliff-like structures of high curvature, often found in the loss functions of deep neural networks, cause large gradient values, which quickly push units out of their linear regions (Pascanu et al., 2013). This can saturate the unit, causing its future gradients to be small and learning slow.

The gradient is a measure of the first-order local rate of change of the loss function. The estimate becomes less accurate when moving further away from the current point. Therefore the gradient should not be used for taking long steps.

One solution is to apply ***clipping*** to the gradient. The gradient can either be

clipped in each dimension separately, essentially truncating the elements of the gradient so that they lie within a predetermined range. Alternatively the norm of the gradient can be clipped, scaling the gradient uniformly if its magnitude is too large.

Another solution for gradient flow problems due to the depth of the network is adding shortcuts to shorten the gradient backpropagation path. A ***residual connection*** (He et al., 2016b) is a shortcut connection that simplifies the task of the layer it is wrapped around. If the layer was originally tasked to learn a function $\boldsymbol{y} = f(\boldsymbol{x})$ between vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ of the same length, the residual connection replaces this with

$$\boldsymbol{y} = g(\boldsymbol{x}) + \boldsymbol{x}. \tag{3.28}$$

The residual $g(\boldsymbol{x}) = f(\boldsymbol{x}) - \boldsymbol{x}$ is typically easier to learn than f. Particularly, if it needs to be close to identity, the weights need only to be pushed towards zero. The highway network (Srivastava et al., 2015) adds a gating mechanism to control the strength of the shortcut. Also the attention mechanism can be used to shorten the path of the gradient.

3.6 Combining multiple models

It is often desirable to combine multiple models for reduced variance and better predictions. Optimally each model's weakness is compensated by the strength of another model. There are two main approaches to using multiple models in concert.

The first is to use the models sequentially in a ***rescoring*** procedure. The first-pass model produces a reduced search space, e.g. an n-best list of candidates. One or more second-pass models score these candidates. The final choice is made from the n-best list based on the combined score of all models. Rescoring is particularly useful when the initial search space is very large, and some of the models are heavy to decode from. The fast first-pass model prunes the search space, while the heavy second-pass model refines the choice. As a downside, if the first-pass model prunes out all good solutions so that they are not included in the top n results, the second-pass model cannot recover them.

The second way uses an ***ensemble*** technique to decode from multiple models simultaneously. In the case of a conditional language model, ensembleing typically interpolates the predictions of the models by averaging the distributions predicted by the individual models into a combined distribution. The average can be computed either of the probabilities (linear average) or their logits (log-linear average). The next symbol is sampled from the combined distribution, and the same symbol is fed into all the models in the ensemble.

4. Subword segmentation

> “Though many things are possible in morphology,
> some are more possible than others.
> (Aronoff and Fudeman, 1976) ”

In order to train a machine learning model on sequential data, the data must be decomposed into basic units or building blocks. For most parametric models, the vocabulary (or lexicon) must be fixed at the beginning of training. For textual data, typical units range from characters to words. Subword units have become a popular choice in recent years.

In rule-based and statistical NLP systems, e.g. phrase-based machine translation, the standard approach is to use tokens separated using white space and punctuation.[1] For these systems, use of subword units has been restricted mainly to morphologically rich languages. The use of subwords has been motivated primarily by the very high out-of-vocabulary (OOV) rates of word-level models (Lee, 2004; Oflazer and El-Kahlout, 2007; Virpioja et al., 2007). Segmentation into characters was proposed initially for translation between closely related languages (Tiedemann, 2009). However, the change of paradigm to neural methods has changed also the practice in vocabulary construction: the standard approach for neural methods is segmentation into subword units (Sennrich et al., 2015). In neural methods, even character (Chung et al., 2016; Costa-jussà and Fonollosa, 2016) or byte (Costa-jussà et al., 2017) segmentation has found use outside of closely related languages. One exception from this trend of small units is unsupervised translation based on cross-lingual mapping of pretrained word embeddings (Artetxe et al., 2018b; Yang et al., 2018).

The ***segmentation method*** is a preprocessing algorithm that divides text into basic units. Several segmentation methods can be applied in sequence, e.g. first applying a rule-based tokenizer to segment the sentence into words, followed by a data-driven subword segmenter.

Much of the work on segmentation has focused on accurately approximating a linguistically motivated segmentation. This focus can be motivated by both ease of intrinsic evaluation, and the belief that theoretically motivated units should perform well in practice. Recent work on unsupervised segmentation is less focused on linguistic fidelity, instead aiming to tune subword segmentations for particular applications. In applications such as NMT, the correspondence of the subwords to linguistic morphemes is not of high importance. The encoder is able

[1] Often called “words”. See Section 2.1 for discussion.

to determine the meaning of the units in context. Therefore the subword segmentation is typically tuned using other criteria, such as the size of the subword vocabulary or the frequency distribution of the units.

Desirable properties for the vocabulary of basic units include high coverage, tractable size, consistency, and ease of extraction. High ***coverage*** ensures that all or at least most of the data is representable using the vocabulary. The unrepresentable parts may be replaced with a special unknown token ⟨UNK⟩. If the proportion of unknown tokens increases, performance deteriorates. For multilingual models, coverage should be balanced between languages. Vocabulary ***size*** affects requirements of both memory (via the number of parameters) and computation (via the length of the sequences and size of sampling distributions). Using large units, e.g. words, results in short sequences, but vocabularies may become intractably large. When using small vocabularies, e.g. characters, memory requirements remain low, but long sequences slow down training, particularly for recurrent networks. ***Consistency*** of segmentation benefits both intra-lingual and cross-lingual generalization. Segmentation should reduce rather than increase ambiguity, resulting in units that are relevant for the modeling task. ***Ease of extraction*** includes the resource requirements (both in terms of computation and data) for both training and applying of the segmentation method. In practical applications, a fast and easy method may be preferable even if accuracy is slightly lower.

The ***granularity*** of the segmentation affects both coverage and size of the lexicon: finer granularity typically means better coverage and smaller lexicon size. However, within the reasonable limits set by the coverage and size, it is much harder to determine the optimal level of granularity. In multilingual models, the right level of granularity for cross-lingual transfer should be used.

This chapter begins with a look at different morphological processing tasks. The focus then narrows on the task of morphological surface segmentation, addressing both evaluation and existing methods for the task. The chapter concludes with the contributions of this thesis to subword segmentation.

4.1 Morphological processing tasks

To produce a natural language sentence, it is necessary to map from an abstract representation of the intended meaning into surface forms, in a way that not only encodes the desired semantics but also achieves syntactic agreement. Morphological generation is one of the last steps of this process. The inverse is required when attempting to understand the communicated utterance. Morphological analysis decodes from surface form to a more abstract representation.

Table 4.1 shows several related morphological tasks that can be described as mapping from one sequence to another. As a formal task, ***morphological generation*** maps from lemma and tags representing the morphological properties of a word to a surface form. ***Morphological analysis*** takes a surface form and yields

Task	Mapping		Example
Generation	$wt \mapsto y$;	$w, y \in \Sigma^*, t \in \tau^*$	take PAST $\mapsto$ took
Analysis	$w \mapsto yt$;	$w, y \in \Sigma^*, t \in \tau^*$	took $\mapsto$ take PAST
Lemmatization	$w \mapsto y$;	$w, y \in \Sigma^*$	better $\mapsto$ good
Reinflection	$wt \mapsto y$;	$w, y \in \Sigma^*, t \in \tau^*$	taken PAST $\mapsto$ took
Segmentation	$w \mapsto y$;	$w \in \Sigma^*, y \in (\Sigma \cup \{+\})^*$	
Canonical			deniability $\mapsto$ deny + able + ity
Surface			deniability $\mapsto$ deni + abil + ity

Table 4.1. Morphological processing tasks. The word forms w and y use the alphabet Σ, while the morphological tags t use the tag set τ.

an abstract representation in the form of the lemma and tags. In ***lemmatization***, the input is an inflected form and the output is the lemma. Lemmatization is a subtask of analysis. The task of ***morphological reinflection*** (see e.g. Cotterell et al., 2016), is a more general variant of morphological generation. Instead of the lemma, one or more inflected forms are given to identify the lexeme, together with the tags identifying the desired inflection. The task is to produce the correctly inflected surface form of the lexeme.

Morphological surface segmentation is the task of splitting words into surface morphs, substrings whose concatenation is the word w. ***Canonical morphological segmentation*** (Kann et al., 2016) instead yields a sequence of canonical morphs. Canonical morphs are standardized segments, the results of undoing any morphological processes modifying the morpheme. Different allomorphs, i.e. different surface morphs corresponding to the same meaning, are thus standardized to the same representation. Canonical segmentation is a compromise between segmentation and analysis, or alternatively a form of analysis in which the tag symbols are chosen to correspond to the surface form of the canonical representative of a morpheme. In this example, the vowel change *"y→i"* is reverted in *"deny"*, and *"+ abil + "* is normalized to *"+ able + "*.

Morphological surface segmentation can be learned effectively in an unsupervised manner. As it does not require disambiguation, it is an easier task than analysis or canonical segmentation, learnable from a smaller amount of data. Annotations for semi-supervised learning are also simple and easy to gather, without requiring deep linguistic knowledge from the annotator. These strengths make morphological segmentation particularly well suited for low-resource languages.

All the discussed tasks operate on individual word forms in isolation. In case of homonyms, methods either produce an exhaustive list of alternatives or only a single alternative, aiming for the most frequent or useful one. In ***morphological disambiguation***, the context in which the word form occurs is used to disambiguate between the alternatives.

To summarize how the tasks relate to each other: In morphological surface seg-

mentation, morph boundaries are inserted to segment a word into stems and concrete morphs, without normalization. This contrasts with morphological analysis and canonical segmentation, which must resolve ambiguity caused by homonymy and allomorphy. Learning to resolve such ambiguity is a more challenging task to learn than surface segmentation. Surface segmentation may be preferred over the other tasks e.g. when used in an application that needs to generate text in a morphologically rich language, such as when it is the target language in machine translation. If surface segments are generated, the final surface form is easily recovered through concatenation.

From now on, the term ***morphological segmentation*** will refer to morphological surface segmentation.

4.2 Evaluation of segmentation

Intrinsic evaluation of segmentations produced by automatic systems against a linguistic gold standard morphological segmentation can be based either on the correct identification of morphs, or on the correct placement of morph boundaries. This may seem like a subtle distinction, but the approaches can give different results for partly correct segmentations. E.g. if *"irreducibly"* is segmented *"irreduc+ibly"* instead of *"ir+reduc+ib+ly"*, there are no correct morphs but one correct boundary.

Oversegmentation occurs when excessive boundaries are introduced inside of the minimal units, resulting in too small units that fail to capture the correct meaning, for example segmenting *"edge"* as *"ed+ge"*. ***Undersegmentation*** is the opposite: it occurs when desired boundaries are not found, resulting in found units that are in fact compounds of multiple minimal units. For example segmenting *"writings"* as *"writ+ings"* results in a unit that is a concatenation of two suffixes, *"ing"* and *"s"*. In a more general sense, over- and undersegmentation can be defined in relation to a segmentation that is optimal for a particular task, instead of using the linguistic morphemes as reference. For example, it can be beneficial for information retrieval tasks to leave derivational suffixes unsegmented, preferring to segment the example above as *"writing+s"* It is possible for a model to simultaneously over- and undersegment different boundaries, even within a single word.

The primary evaluation measures for segmentations used in this work are boundary precision, boundary recall, and boundary F_1-score (Virpioja et al., 2011b). The method combining the use of these three measures is referred to as BPR. The ***boundary precision*** is the proportion of correctly generated boundaries with respect to all generated boundaries, while ***boundary recall*** is the proportion of correctly generated boundaries with respect to the reference boundaries,

$$\text{Precision} = \frac{\text{C(correct)}}{\text{C(proposed)}}; \qquad \text{Recall} = \frac{\text{C(correct)}}{\text{C(reference)}} \tag{4.1}$$

where C is the occurrence count. The ***boundary F_1-score*** equals the harmonic

mean of the boundary precision and recall

$$F_\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}. \tag{4.2}$$

Precision and recall are calculated using macro-averages over the word types in the test set. In the case that a word has more than one annotated segmentation, the one that gives the highest score is taken. As none of the methods evaluated in this work produce a list of alternative segmentations, this approach is motivated. In the case where both the prediction and reference contain multiple entries, the alternatives can be optimally matched using the Hungarian algorithm (Kuhn, 1955; Munkres, 1957), used e.g. by (Virpioja et al., 2011b).

For more closely analyzing the modeling of different aspects of morphology within the framework of boundary evaluation, different boundaries can be tallied separately. The simplest way to accomplish this was used for the first time in Publication I. The test set is divided into subsets according to the patterns of morphs in the reference segmentation: e.g. words without structure (STM), words with a single stem and a single suffix (STM+SUF), uninflected compound words (STM+STM), and so on. Comparing BPR results for these subsets sheds light on how well the method handles words of that type.

Another approach, first used in Publication II, is to compute statistics by classes of boundary rather than classes of word. For undersegmentation, the missing boundaries are categorized according to the pair of morph categories preceding and succeeding the boundary. E.g. if *"matchbox"* is left unsegmented, it is an undersegmentation error of type (STM, STM). For oversegmentation, the extra boundary is categorized according to the category of the morph in which it occurs.

Other methods for intrinsic evaluation of morphological segmentation not used in this work include methods based on an analysis of morph co-occurrence in word pairs from prediction and reference (Kurimo et al., 2009; Spiegler and Monson, 2010; Virpioja et al., 2011b).

Intrinsic evaluations aim to directly compare the predictions against known references. ***Extrinsic*** or indirect evaluations instead evaluate the segmentations based on the down-stream performance when used in an NLP task such as machine translation (See Section 5.3.1), language modeling (Kurimo et al., 2017; Devlin et al., 2019), information retrieval (Kurimo et al., 2010a; Turunen and Kurimo, 2011), automatic speech recognition (Hirsimäki et al., 2006, 2009; Smit et al., 2017), and spoken keyword search (Narasimhan et al., 2014; Singh et al., 2019). Extrinsic evaluation is particularly useful when the optimal representation for a particular application is not known. A downside is that the indirectness of the evaluation diminishes the size of measured differences and introduces noise, increasing the risk of inconclusive results.

4.3 Subword segmentation methods

Finite-state transducers (FST), proposed for morphological analysis in the 1980s (Koskenniemi, 1983), are a commonly used modeling framework for morphological analyzers and generators. For example the Giellatekno project (Moshagen et al., 2013) uses FSTs to model morphology of low-resource languages. In this work, the Omorfi (Pirinen, 2015) FST-based analyzer for Finnish is used. Despite the development tools and computational resources improving greatly in the previous decades, the bottleneck for this rule-based approach is still the large amounts of manual labor and skill that are required (Koskenniemi, 2008).

An alternative data-driven approach, unsupervised morphological segmentation, dates back to the 50's (Harris, 1955), and saw its most active research period in the 2000s and early 2010s. A seminal work by de Marcken (1996) inspired much of the work during this peak of activity. For surveys covering this period, see Hammarström and Borin (2011) for unsupervised methods; Publication II for semi-supervised methods. Zhu (2006) presents a survey on semi-supervised learning in general. I will present a brief survey in the following section, with focus on unsupervised and weakly supervised methods.

The primary distinction for categorizing data-driven segmentation is procedural vs model-based methods. ***Procedural*** methods typically define a heuristic score, such as the letter successor variety (LSV) (Harris, 1955; Hafer and Weiss, 1974; Keshava and Pitler, 2006; Dasgupta and Ng, 2007; Demberg, 2007), and an algorithm that directly applies it for segmentation. LSV is still popular as a heuristic feature for other methods (Ruokolainen et al., 2014; Kurfalı et al., 2017). A related approach involves using language model predictability as a heuristic (Silfverberg and Hulden, 2018).

The alternative to procedural methods is to define a model of morphology, and then fit it using data. ***Model-based*** methods can be categorized in many ways, e.g. into generative vs discriminative models. Probabilistic ***generative*** methods model the probability $P(\boldsymbol{s})$ of generating a sequence of morphs (a word, sentence, or corpus) $\boldsymbol{s} = [m_0, \ldots, m_I]$, as opposed to ***discriminative*** methods that model the conditional probability of the segmentation (boundaries) given the unsegmented data. In unsupervised morphological segmentation, generative models are much more common, but some unsupervised discriminative methods have also been proposed (Poon et al., 2009; Narasimhan et al., 2015; Luo et al., 2017).

Typically generative models perform ***subword lexicon learning***, while discriminative methods perform ***boundary prediction***. Some notable exceptions include the semi-supervised generative boundary prediction method of Spiegler and Flach (2010), and the unsupervised discriminative lexicon learning methods of Poon et al. (2009) and Luo et al. (2017). In supervised learning, there are also some discriminative methods emitting morphs rather than boundaries (Kudo et al., 2004; Cotterell et al., 2015). Eger (2013) presents a fully supervised but not discriminative method, based on exhaustive enumeration with a generative

Markov model.

The minimum description length principle[2] has been an inspiration for many methods (Deligne and Bimbot, 1997; Goldsmith, 2001; Baroni et al., 2002) in addition to the Morfessor family of methods (see Section 4.3.5).

Bayesian models are a very popular choice (Naradowsky and Goldwater, 2009; Can and Manandhar, 2012; Lee et al., 2011; Kurfalı et al., 2017; Dreyer and Eisner, 2011; Snyder and Barzilay, 2008). Typically these methods lack tractable exact solutions, instead relying on Gibbs sampling Markov-chain Monte Carlo for posterior inference. One Bayesian method, Adaptor Grammar, has achieved strong results in unsupervised learning (Johnson, 2008; Sirts and Goldwater, 2013; Eskander et al., 2016, 2019). Adaptor Grammars learn latent tree structures over an input corpus. Formally the method can be described as a hierarchical Dirichlet process that generates a distribution over distributions over trees. Adaptor Grammars can be used to define morphological grammars of different complexity, enabling the user a principled way to exploit linguistic knowledge.

Most data-driven methods model concatenative morphology, with a theoretical basis in item-and-arrangement.[3] Methods based on linking pairs of words with a transformation turning one into the other (Kohonen et al., 2008; Virpioja and Kohonen, 2009; Bernhard, 2009; Narasimhan et al., 2015) could be described as following item-and-process. Neuvel and Fulop (2002) forgoes abstract morphemes entirely in favor of processes. Lavallée and Langlais (2009) base their method on formal analogies between pairs of words. Some methods instead aim to extract paradigms (Snover et al., 2002; Monson et al., 2008; Dreyer and Eisner, 2011).

Methods vary in the limitations they place on the morphological structure. The most limited methods only support a single boundary per word, dividing the stem from the suffixes (Yarowsky and Wicentowski, 2000; Kazakov and Manandhar, 2001; Schone and Jurafsky, 2001; Snover et al., 2002; Monson et al., 2008; Naradowsky and Goldwater, 2009). Others allow slightly more flexibility with a single stem but multiple affixes, most notably Linguistica (Goldsmith, 2001). For languages with extensive compounding and use of linking morphemes, even this limitation is too strict.

The level of supervision varies from unsupervised, via semi-supervised, to fully supervised. As a caveat it should be mentioned that few methods are truly unsupervised, as typically at the very least some hyper-parameters are set using development data or linguistic expertise. It is also possible to turn any unsupervised method into a weakly supervised method by use of data selection. According to Daumé III (2009), semi-supervised methods can be further divided into those starting from an unsupervised method to which the ability to use a small amount of labeled data is added (Poon et al., 2009; Sirts and Goldwater, 2013; Spiegler and Flach, 2010) and those starting from a fully supervised method which is extended to exploit additional unlabeled data (Ruokolainen et al., 2014; Kann et al., 2018; Sorokin, 2019).

[2]See Section 3.3.3 for more information on MDL.

[3]See Section 2.3 for a presentation of morphological theories.

Indirect sources of supervision can also be exploited, including seed knowledge supplied by a human[4] (Yarowsky and Wicentowski, 2000; Wicentowski, 2004; Eskander et al., 2016), and parallel multilingual data (Snyder and Barzilay, 2008; Mermer and Akın, 2010; Naradowsky and Toutanova, 2011; He et al., 2020, and Publication VIII). Cross-lingual transfer can also be exploited in multilingual models without parallel data (Kann et al., 2018; Eskander et al., 2019). There is also a line of research into using semantico-syntactic information extracted from unlabeled corpora using e.g. word embeddings (Schone and Jurafsky, 2001; Narasimhan et al., 2015; Kurfalı et al., 2017; Sakakini et al., 2017; Luo et al., 2017). Both this information and other distributional cues (Can and Manandhar, 2009, 2012; Lee et al., 2011) can e.g. be used for removal of spurious patterns or for paradigmatic clustering.

Recent work on unsupervised segmentation has shifted away from aiming towards a linguistic morpheme segmentation, instead aiming to find optimal subword segmentations for particular applications. The paradigm shift to neural networks has accelerated the process, as neural methods excel at exploiting the full sentence context, while simultaneously benefiting from small vocabularies.

The most prominent of these methods is Byte Pair Encoding (Sennrich et al., 2015, see Section 4.3.3). Other methods include WordPiece (Schuster and Nakajima, 2012; Wu et al., 2016), SentencePiece (Kudo and Richardson, 2018, see Section 4.3.4), and Dynamic Programming Encoding (He et al., 2020). Along the same lines of general sub-word segmentation without regard for morphology lies the use of overlapping character n-gram features in e.g. representation learning (Bojanowski et al., 2017).

Before introducing the contributions of this thesis to subword segmentation, we shall take a closer look at some data-driven segmentation methods proposed for machine translation, or used in this thesis.

4.3.1 Conditional random fields

Conditional random fields (CRF) are graphical models for discriminative structured classification.[5] CRFs are suitable for sequential tagging and segmentation (Lafferty et al., 2001).

For use in discriminative training, morphological surface segmentation can be formulated as structured classification

$$w \mapsto y; \quad w \in \Sigma^k, y \in \Omega^k, k \in \mathbb{N}; \quad \text{e.g.} \quad \textit{uses} \mapsto \textit{BMES} \tag{4.3}$$

where Ω is the segmentation tag set. Different tag sets Ω can be used for segmentation. The minimal sets (used e.g. by Green and DeNero, 2012) only include two labels: Either the beginning (B) or end (E) of segments is distinguished from non-boundary timesteps in the middle (M), resulting in the BM and ME mark-

[4] E.g. lists of affixes or roots in the language.
[5] See Section 3.2.1 for more on CRFs.

ing schemes, respectively. A more fine-grained approach BMES[6] (used e.g. by Ruokolainen et al., 2014) uses four labels. In addition to marking both beginning and end of segments, a special label is used for single-character (S) segments.

Note that there is no need to generate characters from the original alphabet, instead a small tag set Ω is used. A predicted sequence of the tags corresponds unambiguously to a segmentation. The fact that the sequence of boundary decisions is of the same length k as the input has also been made explicit.

4.3.2 Unigram language models for subword segmentation

Several of the generative segmentation methods described in this section apply a ***unigram language model*** to morphological segmentation. The central assumption in the unigram language model is that the morphs in a word occur independently of each other. In an n-gram language model, the history of length $n-1$ is used as conditioning to predict the n:th token. By using unigrams, i.e. setting $n=1$, the length of the history becomes zero. Thus the unigram language model is a zero-order (memoryless) Markov model. The probability of a sequence of morphs decomposes into the product of the probabilities of the morphs of which it consists

$$P(\boldsymbol{s}\,|\,\boldsymbol{\theta}) = \prod_{i=1}^{N} P(m_i\,|\,\boldsymbol{\theta}). \tag{4.4}$$

Note that the Markov model emits morphs that can cover multiple characters. For decoding from such a model given an unsegmented stream of characters, a generalization of the Viterbi (1967) algorithm is needed. The generalized algorithm corresponds to one decoding from a related but different model: a ***hidden Markov model*** with the index of the previous morph boundary as the hidden state[7] and individual characters as observations.

4.3.3 Byte Pair Encoding

Byte Pair Encoding (BPE) was initially proposed as a data compression algorithm (Gage, 1994). Gallé (2019) places BPE in a framework of macro-based compression algorithms. Given a vocabulary size budget, it strives to minimize the encoded length of a corpus by substituting frequently occurring byte pairs with new symbols.

A variant of BPE was applied to neural machine translation (NMT) by Sennrich et al. (2015), and has since then become a standard algorithm for subword segmentation in neural methods for many NLP tasks. While the original algorithm operated on pairs of bytes, as stated in the name of the method, the modern application in NLP instead applies the method to character bigrams.

The algorithm constructs a substitution table, which maps bigrams to symbols.

[6] Also known as BIES, where I stands for internal.

[7] With the span between the previous boundary and current index defining a character multigram.

The table is initialized to include individual characters. Then the most frequent pair of consecutive characters is identified, and added to the table. The procedure is repeated until the desired table size is achieved (or no pair has frequency higher than 1). The substitutions are recursive, meaning that a new symbol introduced by a previous iteration can be used as an element in a new pair. The training procedure is greedy, always adding the most frequent pair without planning ahead. New data can be deterministically segmented by applying the found substitutions in order.

BPE has properties that are desirable for neural NLP systems. The method has only one hyper-parameter, which exactly determines the size of the subword vocabulary. Fine-grained control of vocabulary size is important, as the vocabulary size affects the size of the embedding layers, which contain a large proportion of the model parameters, and also the computationally expensive softmax operation.

Another desirable property is a consequence of greedily joining by frequency, which causes frequent words to be fully joined into single tokens, while rare character sequences remain highly segmented. As a result, the frequency distribution is reshaped in a way that truncates the long tail of rare items. The result approximates using a shortlist of frequent words with subword segmentation for the rare words not on the list.

BPE is a substitution dictionary method, not a probabilistic model. However, the coding bears a similarity to unigram language models in that every subword m_i is encoded individually. From Shannon's source coding theorem (Shannon, 1948) follows that any unigram model provides an implicit compression with code lengths of $-\log_2 P(m_i)$ bits. In the opposite direction, the Kraft–McMillan inequality (Kraft, 1949) proves that any uniquely decodable coding can be associated with a probability distribution, although possibly a deficient one.

In multilingual settings such as translation, a joint BPE vocabulary is typically trained on a concatenation of corpora of the different languages. This improves consistency of segmentation, at a cost of some loss of compactness. If the data is balanced over the languages, the frequent words will be constructed in the early steps of the algorithm for all languages. Joint training is particularly useful for tokens which can be copied from source to target. When such tokens are segmented into the same subwords on both sides, the copying can proceed with a one-to-one subword correspondence. If the segmentations differ, the model must instead combine the subwords into a joint representation, and then generate a different sequence of multiple subwords from that representation.

BPE has been extended in various ways. For example Wu and Zhao (2018) extend BPE with additional features for determining the next merge operation to add, and BPE-dropout (Provilkov et al., 2019) introduces stochasticity which enables the use of BPE with subword regularization.[8]

[8]For subword regularization, see Section 5.5.11.

4.3.4 EM+Prune methods for segmentation

Even with the simplifying independence assumption of the unigram language model, optimizing the model parameters and the vocabulary simultaneously is intractable. Keeping the vocabulary fixed, the parameters of the unigram language model can be optimized using the Expectation Maximization (EM) algorithm.[9] In the E-step, expected observation counts for each of the subwords are computed. In the M-step, the model parameters are updated based on the expected counts.

However, the EM algorithm only updates the expected frequencies for the subwords in the current vocabulary, and can neither expand nor contract the vocabulary. Any subword with nonzero probability in the previous iteration will continue to have nonzero probability after the EM update. Likewise, any subword with zero probability will remain at zero.

To perform subword vocabulary learning, it is therefore necessary to introduce a separate vocabulary pruning step. The vocabulary is initialized to a large seed vocabulary, e.g. the set of frequent substrings. Interleaved with the EM iterations, a pruning step ranks subwords by the estimated change in the loss function if the subword were to be removed.

The methods applying EM+Prune vary in details such as how the loss function is defined, the seed lexicon with which the algorithm is initialized, how the pruning is performed, which variant of EM is used, and which stopping condition is used.

Multigram segmentation

Deligne and Bimbot (1995, 1997) present an unsupervised EM+Prune method that they apply initially to segmentation of sentences into phrases, and later to segmentation of sentences into word-like units. The white space is removed from the sentences before using them as input. While the intended application is different from unsupervised subword segmentation, the method bears a close resemblance to the subword segmentation methods described in this section.

The authors refer to the model as a multigram language model rather than a unigram language model. This naming discrepancy stems from the two views of the model described in Section 4.3.2: either as a Markov chain of morphs or a hidden Markov model of multigrams.

The MAP loss function corresponds to a two-part MDL formulation.[10] The seed lexicon consists of substrings between 1 and 5 characters long, with a frequency threshold applied. The lexicon is pruned using a heuristic removing of multigrams having a very low prior probability. Parameter estimation uses standard EM, and iterations alternating EM and pruning continue until the resulting segmentation converges.

[9] See Section 3.5.1 for expectation maximization.
[10] See Section 3.3.3 for more information on MDL.

Greedy Unigram Likelihood Segmentation

The subword segmentation method of Varjokallio et al. (2013) is particularly designed for use in automatic speech recognition. It applies a multi-phase EM+Prune procedure, with greedy pruning based on unigram likelihood. The seed lexicon is constructed by enumerating all substrings from a list of common words, up to a specified maximum length. Pruning is divided into two phases, which the authors call *initialization* and *pruning*.

The first phase uses a character-level language model to compute the initial probabilities of the subwords. The probabilities are refined by EM, followed by hard-EM. During the hard-EM, frequency based pruning of subwords begins.

In the second phase, parameters are re-estimated using hard-EM. At the end of each iteration, the least frequent subwords are selected as candidates for pruning. For each candidate subword, the change in likelihood when removing the subword is estimated by resegmenting all words in which the subword occurs. After each pruned subword, the parameters of the model are updated. Pruning ends when the goal lexicon size is reached or the change in likelihood no longer exceeds a given threshold.

In short, this method is a maximum likelihood EM+Prune with a heuristic iteration structure, and a heuristic stopping condition based on thresholding the change in likelihood.

SentencePiece

Subword segmentation is potentially ambiguous, even given a particular subword vocabulary. For example, if all individual characters are included in the subword vocabulary, then it is always possible to segment a word fully into characters. If *"the"* is included in the vocabulary, it can either be left unsegmented or segmented *"t ⫩ h ⫩ e"*.

While most segmentation methods strive to limit the segmentation ambiguity, SentencePiece (Kudo and Richardson, 2018; Kudo, 2018) attempts to harness it for regularization. For more on subword regularization, see Section 5.5.11.

SentencePiece applies a unigram language model, in which a word consists of a bag of subwords generated independently of each other. The parameters are estimated using an EM+Prune procedure. As an approximation, it is assumed that when a subword is removed, all its probability mass goes to the subwords in its Viterbi segmentation. At each pruning iteration, a fixed proportion (typically selected to be between 75% and 80%) of the vocabulary is kept, together with all single character subwords. The iteration terminates when the goal vocabulary size is reached.

Even though Kudo (2018) states that the parameters are set to maximize the likelihood, in the reference implementation[11] the maximum a posteriori (MAP) estimate is used. The sparsity inducing prior, called ***Bayesian EM***, is based on

[11] `https://github.com/google/sentencepiece`

approximating a Dirichlet Process using a symmetric Dirichlet distribution prior

$$\boldsymbol{\theta} \sim \text{Dirichlet}_K\left(\frac{\alpha}{K}, \ldots, \frac{\alpha}{K}\right), \tag{4.5}$$

in which the concentration parameter $\boldsymbol{\alpha}$ goes towards zero and the number of categories $\boldsymbol{K}$ goes towards infinity. As a result, the posterior counts for the events $z \in \mathbf{Z}$ are

$$\boldsymbol{\theta}_z = \frac{\exp \Psi(\mathrm{C}(z))}{\exp \Psi(\sum_{z' \in \mathbf{Z}} \mathrm{C}(z'))}, \tag{4.6}$$

where Ψ is the digamma function (Liang and Klein, 2007).

Bostrom and Durrett (2020) find that subword segmentation based on the unigram LM (SentencePiece) is superior to BPE for Transformer based language models.

4.3.5 Morfessor family

Morfessor is a widely used family of morphological segmentation methods, formulated in a generative probabilistic framework with inspiration from the Minimum Description Length (MDL) principle (Rissanen, 1978).[12] While the Morfessor family is language-independent, it has a particular design focus on agglutinative languages with long sequences of morphemes. Unlike earlier MDL-based methods, e.g. Linguistica (Goldsmith, 2001), Morfessor does not restrict the number of stems or affixes in a single word.

Since the original formulation of Morfessor by Creutz and Lagus (2002), the family of methods has seen active development. The main branch of this development and the most direct successor to the original formulation is called Morfessor Baseline. Several variants extending various components of the method have also been introduced. In addition to the Morfessor variants—both previous and novel methods—described in this work, other important variants include Morfessor for segmenting utterances into phrasal constructions (Lagus et al., 2009), and Allomorfessor for modeling allomorphic variation in stems (Kohonen et al., 2008; Virpioja and Kohonen, 2009)

In the following section, a unified formulation of Morfessor is given. It builds on the overview given by Virpioja (2012). As Morfessor is a parametric machine learning method, it consists of three components: model, loss function, and algorithms for training and decoding.

Model. All Morfessor models are generative probabilistic models for segmentation of words, i.e. they define the joint probability $P(\boldsymbol{w}, \boldsymbol{s} | \boldsymbol{\theta})$ of words $\boldsymbol{w}$ and their analyses $\boldsymbol{s}$ given the parameters $\boldsymbol{\theta}$. The structure of the analysis is as a list of morpheme labels $\boldsymbol{s} = (m_1, \ldots, m_I)$, following the item-and-arrangement approach to morphology (see Section 2.3).

Morfessor assumes that the probabilities of words are conditionally independent of each other given the parameters, meaning that no sentence-level or larger

[12] See Section 3.3.3 for more information on MDL.

context is used to disambiguate segmentations. This independence assumption allows factorizing the joint probability of the whole data set as a product of the probabilities of individual words

$$P(\boldsymbol{D} \mid \boldsymbol{\theta}) = \prod_{j=1}^{|\boldsymbol{D}|} P(\boldsymbol{w}_j, \boldsymbol{s}_j \mid \boldsymbol{\theta}). \tag{4.7}$$

The probability of an analysis for a given word is obtained using Bayes' theorem

$$P(\boldsymbol{s} \mid \boldsymbol{w}, \boldsymbol{\theta}) = \frac{P(\boldsymbol{w} \mid \boldsymbol{s}, \boldsymbol{\theta}) P(\boldsymbol{s} \mid \boldsymbol{\theta})}{P(\boldsymbol{w} \mid \boldsymbol{\theta})} \tag{4.8}$$

$$P(\boldsymbol{s} \mid \boldsymbol{w}, \boldsymbol{\theta}) \propto P(\boldsymbol{w} \mid \boldsymbol{s}, \boldsymbol{\theta}) P(\boldsymbol{s} \mid \boldsymbol{\theta}). \tag{4.9}$$

Under the reasonable assumption that a certain analysis $\boldsymbol{s}$ yields only one word form given particular parameters $\boldsymbol{\theta}$, the entire probability mass of the word given the analysis is concentrated to a single word. To express this formally, the word can be named by defining the detokenization function $\phi^{-1}(\boldsymbol{s}, \boldsymbol{\theta})$, which for the models described in this thesis is simply the concatenation of $\boldsymbol{s}$. Using ϕ^{-1} and the indicator function I

$$P(\boldsymbol{s} \mid \boldsymbol{w}, \boldsymbol{\theta}) \propto \mathrm{I}(\phi^{-1}(\boldsymbol{s}, \boldsymbol{\theta}) = \boldsymbol{w}) P(\boldsymbol{s} \mid \boldsymbol{\theta}). \tag{4.10}$$

Thus the Morfessor model is characterized by the probability of the morph sequence $P(\boldsymbol{s} \mid \boldsymbol{\theta}) = P(m_1, \ldots, m_I \mid \boldsymbol{\theta})$. Morfessor Baseline defines this probability using a unigram language model, while other variants make use of more complex models.

The lexicon of used morphs is encoded in the model parameters. While the parameters vary depending on the model, they can be divided into two groups: the morph lexicon L and the grammar G. The parameters in the lexicon can be further divided into those describing the ***form*** of the morph (e.g. the string representation) and those describing the ***usage*** (e.g. the frequency). The grammar (or morphotactic) parameters describe the valid ways of combining morphs to form words.

Loss function. Morfessor training finds a point estimate for the parameters. In most Morfessor methods, the optimal parameters are found using MAP-estimation with an MDL-inspired loss function. The MAP estimate yields a two-part loss function, consisting of the prior (the cost of encoding the lexicon) and the likelihood (the cost of encoding the corpus, given the lexicon).

For most Morfessor methods, if a morph has a non-zero probability given the parameters, it is implicitly considered to be stored in the morph lexicon. As a consequence, the parameter configurations are typically sparse. The priors placed on the model parameters are designed to favor lexicons with fewer, shorter morphs.

The prior for the lexicon is of the form

$$P(\mathrm{L}) = V!\, P(V)\, P\big(\text{properties}(m_i, \ldots, m_V)\big), \tag{4.11}$$

where V is the size of the morph lexicon, and the properties are the form and usage properties characterizing the morph. The factorial term accounts for the $V!$ possible ways of ordering a set of V items. The lexicon is equivalent regardless of the order in which the morphs are encoded. The prior for the lexicon size $P(V)$ is often omitted due to having a negligible effect.

Algorithms. Most Morfessor methods optimize their parameters using a local search procedure. Local search is described in Section 3.5. The search is called local, because only a subset of parameters are optimized in a single search step. The parameters to be optimized depend on the ***neighborhood function***. For Morfessor, the neighborhood function is determined by the units that are reanalyzed in the search step, and the alternative segmentations that are considered. The units can be whole words or substrings of words, and they are not limited to the morphs in the current lexicon. Typically a single search step affects the segmentation of more than one word: e.g. all words with a particular sequence of morphs are simultaneously modified in analogous ways. Even though the change may be beneficial to the score in some words and detrimental in others, each search step is accepted or discarded as a whole.

The initialization and neighborhood function can be modified in various ways. Frequency thresholding (Creutz et al., 2007) omits training words with a frequency below the preset threshold, in order to remove noise such as spelling errors. Originally the frequency thresholding was also proposed as a means of controlling the growth of the lexicon, but this function has been superseded by the weighting hyper-parameter α. Frequency dampening (Virpioja et al., 2011a) controls the effect of word frequencies, by transforming them with a dampening function during initialization. When using the constant function $d(w) = 1$, frequency information is completely removed for ***type-based*** training. Type-based training segments the affixes from frequent inflected forms as much as from less frequent ones, typically giving good results when aiming towards a linguistic segmentation. Using the identity function turns off frequency dampening, recovering ***token-based*** training. Token-based training leaves frequent words unsegmented, while oversegmenting rare words into small but frequent units. This evens out the frequency distribution of the units, which is desirable when used in neural NLP methods. Logarithmic dampening $d(w) = \lceil \ln(1 + \mathrm{C}(w)) \rceil$ is a compromise between type and token-based training. Forced splitting (Virpioja et al., 2013) introduces segmentation boundaries before and/or after certain characters, such as hyphens, apostrophes, and colons. Special characters are generally difficult to segment correctly, motivating this rule-based exception. Forced splitting is not a complete solution to the problem, as special characters share the property with short linking morphemes such as the *"+o+"* in neoclassical compounds such as *"extremophile"*, which are not suited for forced splitting.

Learning setups. Morfessor was initially developed for unsupervised learning. Kohonen et al. (2010) extend Morfessor to semi-supervised learning. Already small amounts of annotated training data were found to substantially improve

the segmentation accuracy. For tuning the model, Kohonen et al. (2010) propose weighting the likelihood with a hyper-parameter α:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \{ -\log \overbrace{P(\boldsymbol{\theta})}^{\text{prior}} - \alpha \log \overbrace{P(\boldsymbol{D} \mid \boldsymbol{\theta})}^{\text{likelihood}} \}. \tag{4.12}$$

The approach is a simple but effective way of using an annotated development set to improve performance. The two-part Morfessor loss strikes a balance between the size of the lexicon (the prior) and the size of the corpus when encoded using the lexicon (the likelihood). These parts of the loss function have opposing optima: the lexicon loss is minimized by a minimal lexicon consisting only of the letters, while the likelihood is minimized by emitting as few large units as possible, i.e. keeping words whole. Therefore, the α hyper-parameter controls the overall granularity of the segmentation learned by the model. High values increase the weight of each emitted morph in the corpus (less segmentation), and low values give a relatively larger weight to a small lexicon (more segmentation).

Another way to use annotated data for semi-supervised training is by adding a third component to the loss function, the cost of encoding the annotations using the lexicon $P(\boldsymbol{D}^{(L)} \mid \boldsymbol{\theta})$, weighted by its own hyper-parameter β. The annotated words are not optimized by the local search steps: the only way that the segmentation of annotated words can change is if multiple alternative annotations are given, in which case a choice between the alternatives is made between epochs. To lower the cost of encoding the annotations, the morphs used in them must be made more frequent. The effect of β is not as straightforward as that of α. It simultaneously affects overall granularity to some extent, and the importance of using the morphs present in the annotations.

To summarize, a member of the Morfessor family can be characterized by the following 9 defining components:

- Model:
  - M1 the probability of an analysis,
  - M2 the detokenization function,
  - M3 the properties of morphs being stored in the lexicon,
  - M4 the parameters of the model grammar,
- Loss function:
  - L5 parameter estimation (ML or MAP),
  - L6 the priors for the model parameters,
- Algorithms:
  - A7 initialization,
  - A8 the training algorithm, and
  - A9 the decoding algorithm(s).

Morfessor Baseline

M1 Morfessor Baseline (Creutz and Lagus, 2002, 2007; Virpioja et al., 2013) applies the unigram language model (Equation 4.4). M2 The detokenization function is concatenation, as in almost all Morfessors. M3 The lexicon is flat, with the form properties encoding the string representation of the morphs, using a character distribution estimated from the data. The usage properties encode the distribution of morph frequencies. M4 In Morfessor Baseline, there are no parameters in the grammar.

L5 Morfessor Baseline finds a point estimate for the model parameters $\hat{\boldsymbol{\theta}}$ using Maximum a Posteriori (MAP) estimation. L6 The currently used priors for the morph frequency and length were introduced by Hirsimäki et al. (2006). The morph frequency prior is based on Rissanen's universal prior for non-negative integers. The implicit exponential morph length prior is achieved by the addition of an end-of-morph symbol. A non-informative prior for the morph frequency distribution is derived using combinatorics: the number of ways that the total token count ν can be divided among the V lexicon items is:

$$P(\tau_1,\ldots,\tau_V \mid V,\nu) = 1/\binom{\nu-1}{V-1}. \tag{4.13}$$

A7 Morfessor Baseline is initialized by setting the analyses to whole words.[13]
A8 The local search training algorithm is based on a ***recursive splitting*** of morphs into two parts. The algorithm loops over words in the corpus in a random order, and looks for an optimal segmentation into exactly two parts. If no segmentation improves the loss function, the recursion terminates. If the string is segmented, the algorithm descends recursively into the substrings. As Morfessor Baseline uses a unigram language model, the optimal analysis of a substring is not context dependent. The recursive splits are stored in an acyclic graph with words as top nodes, morphs as leaf nodes, and intermediary recursion states as internal nodes. This graph structure is used only during training to ensure that the local search modifies in a consistent way all words with a particular sequence of morphs, and can be discarded once training is complete. Iterations over the words in the corpus are repeated until the search converges to a local optimum. Note that in contrast to BPE, which proceeds bottom-up constructing subwords starting from single characters, Morfessor Baseline starts from the whole word and can thus be considered a top-down algorithm. The training algorithm of the Morfessor Baseline method is described in more detail by Creutz and Lagus (2005b) and Virpioja et al. (2013).

An alternative training algorithm has also been implemented. In ***Viterbi training***, each word is individually resegmented to the most likely segmentation given the current model parameters. This approach is also known as *hard-EM*. Viterbi training is not able to perform analogous changes in multiple words in a single search step. Also, no new subwords can be introduced during Viterbi training.

[13] Virpioja et al. (2013) experimented with random initialization, with inconclusive results.

For these reasons, the algorithm is most suited for use as a fine-tuning algorithm after training with the recursive search.

Morfessor CatML and CatMAP

The simplifying assumption of context independence which enables the unigram language model in Morfessor Baseline does not actually hold for natural language. E.g. a valid suffix can not necessarily be used as a prefix, but the unigram language model is unable to assign it a different probability in the two cases. This can lead to segmentation errors such as *"*s+ team"*. In order to impose morphotactic rules, a hidden Markov model with morph categories can be used for the probability of a morph sequence. Morfessor Categories-ML (Creutz and Lagus, 2004) (CatML) and Categories-MAP (Creutz and Lagus, 2005a) (CatMAP) are two Morfessor variants taking this approach. Each morph is assigned one of four categories: prefix (PRE), stem (STM), suffix (SUF), or non-morpheme (NON).

M1 A hidden Markov model is used for assigning probabilities to analyses,

$$P(\boldsymbol{s}\,|\,\boldsymbol{\theta}) = P(c_1\,|\,\_)\prod_{i=1}^{|\boldsymbol{s}|}\left[P(m_i\,|\,c_i)P(c_{i+1}\,|\,c_i)\right]P(\_\,|\,c_{|\boldsymbol{s}|}), \tag{4.14}$$

where _ indicates both the word boundary symbol and its state. The independence assumption made by this model is that there is no lexical dependence between consecutive morphs, given the sequence of morph categories. As such it cannot model all morphotactic constraints, e.g. the order in which affixes marking certain morphological categories should be used, or phonotactic constraints such as which allomorph should be selected based on sounds in previously used morphs. The HMM has morph categories as hidden states and morphs as observations. In addition to the four categories mentioned previously, a state for the word boundary is also needed. The word boundary state can only emit the boundary marker.

M3 CatMAP also introduces a hierarchical lexicon, intended to reduce the undersegmentation of words with frequently occurring morph sequences. In a hierarchical lexicon, morphs already present in the lexicon can be reused when encoding new morphs. The form properties of a morph depend on whether it is a new, spelled out morph, or a structured morph built from existing submorphs,

$$P\big(\text{form}(m_i)\big) =$$

$$\begin{cases} \big(1-P(\text{substruct.})\big)\prod_{j=1}^{|m_i|}P(\sigma_{ij}) & \text{Spelling out} \\ P(\text{substruct.})P(c_{i1}\,|\,m_i)P(m_{i1}\,|\,c_{i1})P(c_{i2}\,|\,c_{i1})P(m_{i1}\,|\,c_{i1}) & \text{Substructure} \end{cases} \tag{4.15}$$

The usage properties must contain information that allows the estimation of the probability distributions assigning categories to morphs. In the HMM, this is defined by the emission probabilities $P(m\,|\,c)$. For estimation, it is more convenient to define $P(c\,|\,m)$ and reverse the conditioning using Bayes' theorem.

$$P(m\,|\,c) = \frac{P(c\,|\,m)P(m)}{P(c)} \tag{4.16}$$

The conditional category distribution is based on two intuitions: stems are assumed to be generally longer than affixes, while affixes are assumed to co-occur with many different stems. Thus the morph is more likely to be an affix if it has a neighbor that is difficult to predict. For suffixes the relevant neighbor is the predecessor, while for prefixes it is the successor. The unpredictability is measured using perplexity in the current segmentation of the training data. Thus three usage properties per morph are required: frequency, left perplexity, and right perplexity. The length in characters does not need to be encoded separately as it is already computable from the encoded string representation. A probability distribution is computed from the length and perplexities via sigmoidal thresholding, assigning of probability to the non-morpheme class, and normalization.

M4 The model grammar consists of a fixed number of parameters depending only on the number of categories. These parameters characterize the transition probabilities of the HMM. They are estimated using maximum likelihood with restrictions forcing some transitions to have zero probability: a suffix may not directly follow the initial word boundary or a prefix, and a prefix may not be directly followed by a word boundary.

L5 As CatML is a maximum likelihood method, it lacks the inherent balance between lexicon and corpus losses. To avoid the lexicon size growing uncontrollably, it requires heuristic restrictions. These restrictions are implemented as constraints on the local search. CatMAP on the other hand uses maximum a posteriori estimation of the model parameters. L6 However, the morph occurrences present in the hierarchical lexicon prevent a clearcut MDL-based factorization of the cost function into two parts relating to the lexicon and corpus respectively. While the parameters have clearly defined priors, the use of $P(m|c)$ during the encoding of the lexicon causes the contributions of lexicon and corpus during estimation to be mixed. The lack of this factorization prevents the use of the α-weighting scheme for semi-supervised training proposed by Kohonen et al. (2010).

A7 CatML and CatMAP are initialized from a Morfessor Baseline segmentation. As the segmentation is initially not tagged by morph categories, the emission and transition probabilities are first estimated using Viterbi-search (a.k.a. hard-EM) without changing the segmentation. A8 CatML applies only two steps which are not repeated: splitting into already existing morphs to remove redundant morphs, and removal of non-morphemes by joining with their neighboring morphs. CatMAP uses a local search with three alternating search steps—SPLIT, JOIN, and RESEGMENT—repeated for a fixed number of iterations. In the SPLIT operation, all occurrences of a particular morph are reanalyzed by trying all allowed two-morph substructures. In the JOIN operation, pairs of consequent morphs are considered for joining. As all occurrences of the new morph are tagged the same way, the pair is only considered for joining if the surrounding context is consistent with the new category. In the RESEGMENTATION operation, each word form is completely reanalyzed using Viterbi search. At the end of training, the non-morphemes are removed by collapsing any structured morph containing

submorphs tagged as non-morphemes. A9 For decoding, the Viterbi search is adjusted to account for the categories, by expanding the state space to be the Cartesian product of previous morphs and previous categories. This also increases the time complexity from $O(|\boldsymbol{w}|^2)$ in the case of a unigram language model to $O(|\boldsymbol{w}|^2 K^2)$ for an HMM model with K categories.

4.4 Contributions to subword segmentation

In the following section, the contributions of this thesis to the field of subword segmentation are presented. Methods are ordered by the Publications in which they are presented, rather than chronologically. The focus lies on describing the range of ideas that were explored. Only intrinsic evaluations are discussed here; for machine translation evaluations, see Section 5.5.

4.4.1 Morfessor FlatCat

Morfessor FlatCat[14] is presented in Publication I and applied in Publications II, III, IV, VII, and X. The HMM morphotactics of Morfessor FlatCat are based on Morfessor CatMAP (Creutz and Lagus, 2005a), but the use of a flat morph lexicon enables using the semi-supervised training approach proposed by Kohonen et al. (2010). Morfessor FlatCat is designed for and excels at semi-supervised training, but the lack of hierarchy in the lexicon is a detriment in fully unsupervised training.

M1 The probability of an analysis is given by Equation 4.14, the same HMM morphotactics previously used in CatML and CatMAP. The benefit of the HMM morphotactics is increased sensitivity to the context in which the morph occurs, which improves the precision of the segmentation by suppressing spurious segmentation. For more discussion on the benefits of the HMM, see Section 4.3.5. M3 Morfessor FlatCat uses a flat lexicon, in contrast to the hierarchical lexicon in CatMAP. A hierarchical lexicon allows building new morphs out of other morphs already in the lexicon, while in a flat lexicon each morph is represented directly as a string of letters.

$$P\big(\text{form}(m_i)\big) = \prod_{j=1}^{|m_i|} P(\sigma_{ij}) \qquad \text{Spelling out} \qquad (4.17)$$

Longer morphs are more expensive to add to the lexicon, as each letter must be encoded separately. Not considering the effect of loss function weighting, the lack of hierarchy leads to a tendency to segment slightly more, evidenced by higher recall but lower precision of boundaries, as seen in Tables 4.3 (English) and 4.4 (Finnish), but not 4.5 (Turkish). The benefit of the flat lexicon comes from the factorization of the loss function. In a hierarchical lexicon, morphs are emitted

[14]Software available at `https://github.com/aalto-speech/flatcat`.

both in the lexicon and the corpus. When using a flat lexicon, all morph occurrences are in the encoded corpus. This enables the factorization required for the weighting of the loss function components seen in Equation 4.12. The loss function weighting allows tuning to compensate for the general tendency of unsupervised methods to undersegment. Direct control of the overall segmentation granularity is also useful for applications: in Publication VII, this tuning ability was used to increase segmentation consistency for statistical machine translation.

M4 L6 The usage properties of morphs, and the parameters of the model grammar are the same as in Morfessor CatMAP. The way in which the conditional category distributions $P(c|m)$ are computed from the length and neighbor perplexities of morphs also follows CatMAP.

L5 Morfessor FlatCat is trained using α-weighted MAP-estimation, with the possibility of semi-supervised training. The neighbor perplexities are model parameters that are heavy to compute exactly during the local search. To speed up learning, the perplexities are estimated from the current analysis only between iterations. As new morphs are introduced during the search operations, their perplexities need to be estimated. When a new morph is introduced by joining two existing morphs, it can inherit left and right perplexity from the first and second submorph, respectively. To improve the estimate, the number of contexts can be thresholded to at most the number of occurrences for the new morph. When a morph is split into two submorphs, the first and second submorph can inherit from the old morph the left and right perplexity, respectively. The other two perplexities are set to 1, making the assumption that the new morphs do not occur in any other contexts. At the end of the epoch, the perplexities estimated in this way are replaced with estimates based on counts from the current analysis.

A7 Training is initialized from a Morfessor Baseline segmentation in the same manner as Morfessor CatMAP. A8 The training algorithm is a local search with alternating search steps. In contrast to CatMAP, which uses a hard-coded iteration structure, FlatCat offers fine-grained control of the iteration structure. Training proceeds until convergence rather than a predetermined number of epochs. The order of search operations can be freely selected, and operations can be easily added or removed.

A new search operation called SHIFT is introduced. SHIFT operates on a pair of consequent morphs, attempting to replace them with another pair in which the boundary has been shifted forward or back within a defined window. E.g. if the current segmentation is *"imp+atient"*, the SHIFT operation allows replacing it with *"im+patient"* in a single search step. Even though limits are placed on the neighborhood function[15] to avoid generating undesirable changes, using the SHIFT operation is not always beneficial. Segmentation quality deteriorating when the search is made more flexible could be an indication of model error.

In Morfessor CatMAP, the morph(s) resulting from a search step receive the same tags in all instances. The selected categories may not be permitted in all

[15]The boundary can be moved at most 2 letters, and the resulting morphemes must be at least 2 letters long.

contexts, e.g. *"e"* and *"d"* can be joined into *"ed/*SUF*"* in *"us/*STM ⧺ *e/*NON ⧺ *d/*NON*"* but not in *"e/*NON ⧺ *d/*NON ⧺ *ict/*SUF*"* as the latter would lead to the forbidden category sequence of a word starting with a suffix. In CatMAP the solution is to limit the scope of the search step by adding context sensitivity: e.g. units in the beginning of words are treated separately from units at the end. In constrast during Morfessor FlatCat search, each word is individually retagged, ensuring that category sequences are both permissible and optimal. The context sensitivity in the neighborhood function was kept, although it is no longer necessary to avoid illegal category sequences. The operators are not fully symmetrical, because the unit sizes of the targets vary: splitting targets a single morph, joining and shifting targets a context-sensitive morph bigram, and resegmenting targets a word form. As postprocessing at the end of training, the non-morphemes are removed by joining with their neighboring morphs. A9 The generalized Viterbi decoding algorithm is identical with the one used in CatMAP.

4.4.2 Morfessor EM+Prune

The greedy local search of Morfessor Baseline is sensitive to initialization, and may converge to suboptimal solutions, particularly in the case of unsupervised learning of small subword lexicons for use in neural NLP. In Publication V, Morfessor EM+Prune[16] replaces the search algorithm of Morfessor Baseline with an improved algorithm based on Expectation Maximization and pruning a seed lexicon of frequent substrings. The training algorithm draws inspiration from SentencePiece (see Section 4.3.4). Morfessor EM+Prune is aimed for the unsupervised setting, tuning for small vocabularies. As it retains the uncertainty of segmentations by estimating expected occurrence counts rather than defining a single current segmentation for each word, the method is particularly well suited for the sampling of alternative segmentations required by subword regularization.

Many defining characteristics are retained from Morfessor Baseline. M1 The probability of an analysis is given by the unigram language model, M2 the detokenization function is concatenation, and M4 there are no parameters in the model grammar. M3 The properties of morphs are similar to Morfessor Baseline. In Morfessor EM+Prune, morphs are explicitly stored in the lexicon, and morphs are removed from the lexicon only during pruning. This differs from Morfessor Baseline, in which a morph is implicitly considered to be stored in the lexicon if it has non-zero count. The usage properties are real-valued expected counts, rather than natural numbers counted from the current analysis.

L5 The parameters are estimated using MAP combined with an implicit Dirichlet Process prior called ***Bayesian EM*** (Liang and Klein, 2007). L6 The priors placed on the lexicon of the Morfessor model must be slightly modified for use with the EM algorithm. The standard Morfessor Baseline prior is used during the pruning phase. During the EM parameter estimation, the prior for the morph form properties is omitted. It has no effect on the parameter estimation, as the

[16]Software available at `https://github.com/Waino/morfessor-emprune`.

morph lexicon remains constant. The switch from counts to real-valued expected counts poses a problem for the frequency distribution prior. Recall[17] that the Morfessor Baseline frequency distribution prior is derived using combinatorics from the number of possible assignments of the total count to morphs in the lexicon. When using real-valued expected counts, there are infinite assignments of counts to parameters, making the prior not theoretically motivated. Despite this, the prior is retained with the minimal modification of rounding the numbers to the nearest integer, in order to use EM+Prune as an improved search to the original loss function.

A7 The seed lexicon for initializing Morfessor EM+Prune consists of the N most frequent substrings, with optional prepruning of redundant subwords. Forced splits before or after certain characters (e.g. hyphens, apostrophes, and colons) may also be enforced by prepruning the seed lexicon. As Morfessor EM+Prune cannot introduce new subwords during training, pruning out morphs spanning over a mandatory split point is enough to ensure that the restriction is enforced. A8 Morfessor EM+Prune implements a new training algorithm based on alternating parameter estimation using the EM algorithm and lexicon pruning. The EM phase of each training iteration consists of three EM sub-iterations. The model parameters are replaced with updated expected occurrence counts, given the previous best parameters. In the following pruning phase, the subwords in the current lexicon are sorted in ascending order according to the estimated change in the loss if the subword is removed. Multi-character subwords are removed until either of the pruning stopping criteria are met: either the estimated loss starts rising, or the pruning quota for the iteration is reached. A9 For finding the optimal segmentation of new data, or an n-best list of alternatives together with their probabilities, the generalized Viterbi search of Morfessor Baseline is applicable. For use in subword regularization (Kudo, 2018), alternative segmentations must be sampled. Sampling from the full data distribution is implemented using the forward-filtering backward-sampling algorithm (Scott, 2002). A faster alternative is to approximate the distribution by the n-best list, which can be cached for reuse.

SentencePiece can only be trained to produce a subword lexicon of predetermined size. In contrast, Morfessor controls the final lexicon size using the tuning hyper-parameter α (Equation 4.12). To reach a subword lexicon of a predetermined size while using the Morfessor prior, Morfessor EM+Prune implements a novel automatic tuning procedure. When ranking the subwords for the pruning step, the estimated change in prior and likelihood are computed separately for each subword. For the majority of subwords, the estimated changes have opposing signs: removal of the subword decreases lexicon cost while increasing corpus cost. Subwords whose removal reduces both loss components are always removed regardless of α, while if both components are positive, the subword will not be removed. For other subwords, it is possible to determine the exact value of α

[17] From Section 4.3.5 L6

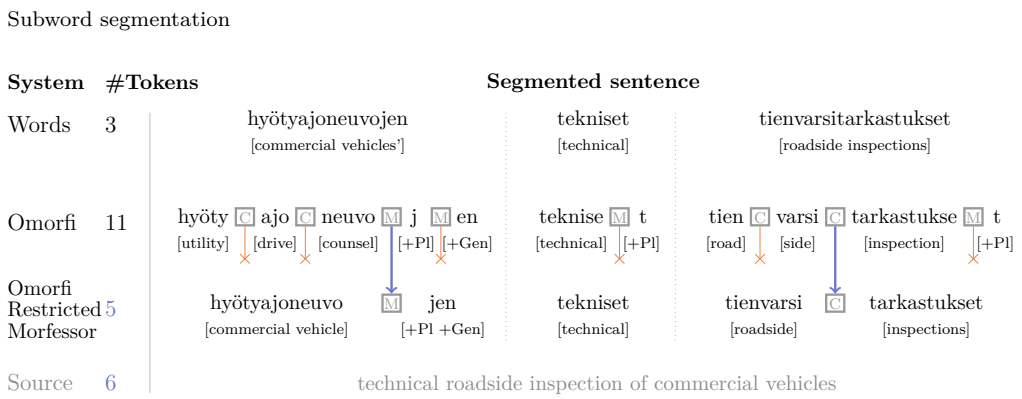


Figure 4.1. Worked example of the Omorfi-restricted Morfessor (ORM) segmentation.

that would balance the changes in the loss components, causing pruning to stop at that point. Arranging the values of α makes it possible to compute the value that gives exactly the desired size of lexicon after convergence of the pruning. As retraining the parameters may cause the estimated threshold α to change, the automatic tuning is repeated before each pruning phase.

Publication V shows that the Morfessor EM+Prune algorithm reduces search error during training, resulting in models with lower Morfessor loss. When segmentation output is compared to linguistic morphological segmentation, the lower losses result in improved accuracy.

4.4.3 Hybridizing rule-based and data-driven segmentation

A linguistically accurate morphological segmentation does not have an optimal granularity for machine translation.[18] For SMT, accurately segmenting chains of affixes is often oversegmentation, while for NMT, keeping rare stems whole is undersegmentation. Data-driven segmentation is able to exploit frequency information to alleviate both of these problems. However, linguistic segmentation contains valuable information from the perspective of language-internal segmentation consistency.

Restricted Morfessor, presented in Publication VI, aims to hybridize the advantages of rule-based segmentation reaching high linguistic fidelity, and tunable data-driven segmentation. The segmentation is tuned from parallel data, aiming for a similar granularity on both sides of the language pair. While restricted Morfessor could be used in combination with any other segmentation method, in the experiments the segmentation tool from Omorfi (Pirinen, 2015) for Finnish was used. This combination is called Omorfi-restricted Morfessor (ORM).

M1 Restricted Morfessor follows the Morfessor Baseline method, but additionally restricts the possible set of segmentation boundaries to those between linguistic morphs. That is, the segmentation method may decide to join any of the linguistic morphs, but it cannot add new segmentation boundaries to known linguistic morphs. Analyses violating the restrictions have zero probability. In

[18] See Section 5.3.1 for discussion.

practice the restrictions are enforced by pruning the search space. A7 Training is initialized with unsegmented words augmented with the information of allowed potential segmentation locations. Instead of this top-down approach, it would be possible to proceed bottom-up by starting with words maximally segmented and allowing Morfessor to recombine the morphs into larger units. A drawback of the bottom-up approach is that it is not obvious how the training-time data structure of Morfessor Baseline should be optimally initialized. E.g. a right-branching tree could be used, but such an initialization may result in convergence to a poor local optimum in the presence of restrictions. A8 Recall[19] that the training-time data structure of Morfessor Baseline contains intermediary nodes representing substrings shared by several word forms. It is possible for two or more word forms to have different restrictions on the same substring, causing some of the restrictions to be violated during training with the recursive algorithm. Full enforcement of restrictions can be ensured by applying the recursive algorithm only for the two first epochs, and then switching to Viterbi training. As Viterbi training resegments each word individually, restrictions apply with the correct scope. In practice the effect is small, as in the experiments only a very small proportion of restrictions were violated during recursive training.

4.4.4 Cognate Morfessor

Publication VIII addresses target–target consistency in the asymmetric-resource multilingual translation task.[20] The method focuses on improving the consistency of morphological segmentation for cognate pairs,[21] to enable the use of cross-lingual transfer to improve the learned subword representations. If segmentation decisions are consistent between the high- and low-resource target languages, the units in the low-resource language can better benefit from the contexts of their correspondents in the high-resource language.

Cognate Morfessor[22] is a multilingual variant of the Morfessor method. It uses automatically extracted cognates paired between the two languages, and fuzzy matching to link cognate morphs. It builds on reuse of components from the Morfessor Baseline model, with modifications to the training algorithm enforcing the cross-lingual restrictions.

The data is arranged into pairs of words, one from each target language. The analysis is thus a pair of morph sequences $(\boldsymbol{s}, \boldsymbol{t}) = \big((s_1, \ldots, s_I), (t_1, \ldots, t_J)\big)$. M2 The detokenization function also returns a pair of words, concatenating both

[19] From Section 4.3.5 A8

[20] Consistency types were introduced in Section 1.1. For more on the asymmetric-resource setting, see Section 5.3.2.

[21] For a discussion on cognates, see Section 2.4

[22] Software available at `https://github.com/Waino/morfessor-cognates`.

languages separately. M1 The probability of an analysis

$$P(\boldsymbol{s},\boldsymbol{t}) = \begin{cases} \prod_{i=1}^{I} P(s_i)\,P(t_i)\,P(\mathfrak{e}_i) & \text{if } |\boldsymbol{s}| = |\boldsymbol{t}| \\ 0 & \text{if } |\boldsymbol{s}| \neq |\boldsymbol{t}| \\ \prod_{i=1}^{I} P(s_i) & \text{if } \boldsymbol{t} = \emptyset \\ \prod_{i=1}^{I} P(t_i) & \text{if } \boldsymbol{s} = \emptyset \end{cases} \tag{4.18}$$

is defined based on a 1:1 alignment between the morph sequences. Analyses with different number of morphs are not allowed. The string edit operations $\mathfrak{e}_i$ transforming the morph s_i to t_i are based on the Levenshtein (1966) algorithm. One side of the pair is allowed to be missing, in order for words for which no cognate pair was found to be used during training.

L5 The MAP loss function

$$\mathrm{L}(\boldsymbol{\theta},\boldsymbol{D}) = \begin{array}{|c|c|c|} \multicolumn{1}{c}{\text{HRL target}} & \multicolumn{1}{c}{\text{LRL target}} & \multicolumn{1}{c}{\text{edits}} \\ \hline -\log P(\boldsymbol{\theta}_{\boldsymbol{s}}) & -\log P(\boldsymbol{\theta}_{\boldsymbol{t}}) & -\gamma \log P(\boldsymbol{\theta}_{\mathfrak{e}}) \\ \hline -\alpha \log P(\boldsymbol{D}_{\boldsymbol{s}} \mid \boldsymbol{\theta}_{\boldsymbol{s}}) & -\alpha \log P(\boldsymbol{D}_{\boldsymbol{t}} \mid \boldsymbol{\theta}_{\boldsymbol{t}}) & -\alpha\gamma \log P(\boldsymbol{D}_{\mathfrak{e}} \mid \boldsymbol{\theta}_{\mathfrak{e}}) \\ \hline \end{array} \tag{4.19}$$

divides both lexicon and corpus coding losses into three parts: one for each language ($\boldsymbol{\theta}_{\boldsymbol{s}},\boldsymbol{D}_{\boldsymbol{s}}$ and $\boldsymbol{\theta}_{\boldsymbol{t}},\boldsymbol{D}_{\boldsymbol{t}}$) and one for the edits transforming the cognates from one language to the other ($\boldsymbol{\theta}_{\mathfrak{e}},\boldsymbol{D}_{\mathfrak{e}}$). A new weight γ is introduced for the edits, with value set to 10. The intuition for encoding the edits is that the changes in spelling between the cognates in a particular language pair is regular. Coding the differences in a way that reduces the cost of making a similar change in another word guides the model towards learning these patterns from the data. L6 The priors for the model parameters follow Morfessor Baseline, with the modification that each model component has its own copy of the priors.

The Levenshtein (1966) algorithm is a dynamic programming algorithm for finding the minimal edits transforming one string into another. The edit operations are insertions, deletions, and substitutions of individual characters. There are variants of the algorithm that omit substitutions or add transpositions. ***Levenshtein distance*** between two strings is defined as the minimum number of atomic character edit operations required to transform one into the other.

M3 Each of the languages has its own subword lexicon. Subword lexicons for both languages are equivalent to Morfessor Baseline models, with form properties encoding the string representations and usage properties encoding the morph frequencies. The edit lexicon is of the same type as the other two lexicons. The strings stored in the edit lexicon consist of a pattern and replacement, divided by a separator character. The set of string edits minimizing the Levenshtein distance are found. Edits that are immediately adjacent to each other are merged. Modeling of sound length change is improved by extending the edit in both languages to cover the neighboring unchanged character. The lengthening is only applied if one side of the edit consists of the empty string ϵ, and the other contains another instance of character representing the sound being lengthened or shortened. E.g. the edit transforming short *"a"* to long *"aa"* is encoded as *"a→aa"* instead of *"ϵ →a"*. Edits with an empty string on one side are undesirable, as the empty

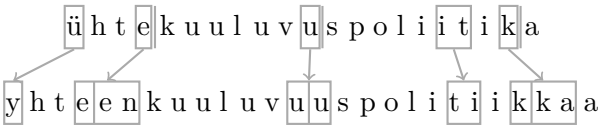


Figure 4.2. Edits found by the Cognate Morfessor algorithm to map between a cognate pair in Estonian and Finnish. The second and third edits demonstrate vowel lengthening, and the last includes a doubled consonant. The fourth edit has the same Levenshtein cost as the pair of vowel shortening *"i→ii"* and lengthening *"ii→i"* that would be correct.

context matches everywhere, making overuse possible. Figure 4.2 shows the edits in an example pair of words.

The restriction that words must have an equal number of morphs is not always a good fit for language use. E.g. Finnish suffix *"+n"* marking the genitive frequently lacks a correspondence in Estonian. To maintain the equal number of morphs in both languages, either the Finnish genitive must be undersegmented, or worse, the Estonian stem oversegmented. In this case it may be preferable to allow a single extra morph to be added at the end of words, by introducing an end of word symbol that may be segmented like an ordinary character. E.g. *"silma→silmän"* can be segmented as *"silma+ϵ"* and *"silmä+nϵ"*. In post-processing, the empty morph is removed from the Estonian analysis.

M4 While the grammar is more complex than in Morfessor Baseline, it can still be considered parameter-free. The restriction that cognate pairs must be segmented into the same number of morphs can be imposed without needing parameters. While the edit operations could be viewed as a part of the grammar, it is more descriptive to place them in the lexicon, considering that they are of the same form as the morph lexicons. A7 Cognate Morfessor is initialized with pairs of whole words (potentially empty), one for each language. A8 The restriction requiring both sides to be segmented into the same number of morphs needs to be imposed during training. To maintain the condition, both sides of the pair are reanalyzed simultaneously. The recursive splitting procedure must split the n-th morph on both sides into two parts, or leave both unsplit. Figure 4.3 visualizes the local search neighborhood for the word pair *"loo+mise+ks"* and *"luo+mise+ksi"*, when reanalyzing the central morph pair. If *"mise"* is split into *"mi+se"* on both sides, no edits are required. Deciding to split e.g. into $(($*"m+ise"*, *"mis+e"*$)$, $($*"ϵ→is"*, *"is→ϵ"*$))$ would require adding two edits to the edit lexicon. Splits resulting in an empty morph on either side are not considered. A9 When decoding, only one side is available. This enables use of the exact same generalized Viterbi as in Morfessor Baseline. Edit operations need not be considered during decoding.

4.4.5 Semi-supervised neural segmentation

Semi-supervised sequence labeling is an effective way to train a low-resource morphological segmentation system when the aim is a linguistic segmentation, and even a small amount of labeled data is available. In this setup, morphological

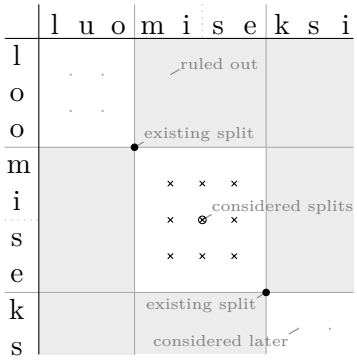


Figure 4.3. Bilingual recursive splitting in Cognate Morfessor.

segmentation is viewed as a sequence labeling task: the boundary between each character is labeled with the information required to determine if it is a morph boundary or not.

Semi-supervised training can be used to hybridize generative and discriminative training methods. The idea is that the generative model is used to find statistical patterns in the large unannotated data. Encoding the decision of the generative model as features for the discriminative model allows it to exploit the found patterns. At the same time, the process frees up capacity of the discriminative model for learning to determine when the predictions of the generative model are reliable. Essentially the discriminative model only needs to learn to correct the mistakes of the generative model.

Ruokolainen et al. (2014) present an approach for combining generative features from Morfessor for discriminative training of a conditional random field (CRF). The success of neural methods in high-resource morphological segmentation (e.g. Wang et al., 2016) motivates an attempt in Publication IV to apply a similar feature set enrichment approach with a neural sequence tagger,[23] for use in a low-resource setting. Morfessor FlatCat (see Section 4.4.1) is used as the generative segmentation method.

Morfessor FlatCat is an especially good choice for producing features for use in discriminative training. Discriminatively trained methods excel in modeling of suffixation, but have difficulty learning to segment the boundaries between stems in compound words (See Publication II). HMM morphotactics improves the segmentation of compound words, by allowing an increase in the overall level of segmentation without causing oversegmentation of stems.

A factored input representation is suitable for use with a neural system. The FlatCat segmentation decision and predicted morph category label are independently embedded. These factor embeddings are concatenated to the character embedding. The morph category labels included in the human annotations (see Section 4.4.6) enable the use of a simple target-side multi-task setup (compare to Section 5.5.7) to predict them in addition to the segmentation boundaries. The

[23]Software available at `https://github.com/Waino/OpenNMT-py/tree/same_length_decoder`.

Subset	Version 1	Version 2
Development	100	199
Training	643	1044
of which actively selected	346	584
of which randomly selected	311	500
Testing	357	796

Table 4.2. The number of annotated word forms in the two versions of the North Sámi morphological segmentation data set. As the same word form can be selected by both random and active selection, the two subsets of the training set are not disjoint.

output vocabulary is extended to cover all combinations of segmentation decision and category label. This is feasible due to the very small number of segmentation labels (BMES + end symbol) and morph categories (STM and SUF), resulting in a combined output vocabulary of 10 labels.

A two-step semi-supervised training procedure is used. The training data consists of a large unlabeled set, and a smaller labeled training set. The labeled training set is further divided into two parts. The generative model (Morfessor FlatCat) is trained in a semi-supervised fashion using the first part of the labeled training set together with the unlabeled data. The words in the second part of the labeled training set are segmented using the generative model. Now the second part of the labeled data includes two segmentations associated with each word: predicted and gold standard. A discriminative model is then trained on the second part of the labeled training set. In the second step, the predictions of the generative model are fed into the discriminative model as augmented features, while the gold standard segmentation is used as the target sequence. The decoding is also a two-step procedure: first the words of interest are fed through the generative model to produce the features. The final segmentation can then be decoded from the discriminative model.

Publication IV argues that segmentation is best formulated as a tagging problem, not a sequence-to-sequence problem. Seq2seq methods are easy to apply, as you can often take e.g. existing neural machine translation software and train it with appropriately preprocessed data. However, arbitrary length sequence-to-sequence transduction is not the optimal formulation for the task, as the output vocabulary and number of parameters are unnecessarily large, and small data sets may not be sufficient for learning to copy the correct characters from input to output. The model is allowed to predict any symbol from its output vocabulary, although only two symbols are valid at any given timestep: the boundary symbol or the actual next character. Kann et al. (2018) apply the seq2seq model for low-resource morphological segmentation, and address this problem by adding an auxiliary autoencoder task.

The proposed neural sequence tagger uses a simplified architecture that is easy to implement by modifying existing NMT software. The encoder is a standard single-layer bidirectional LSTM. The decoder is a single-layer LSTM, which takes

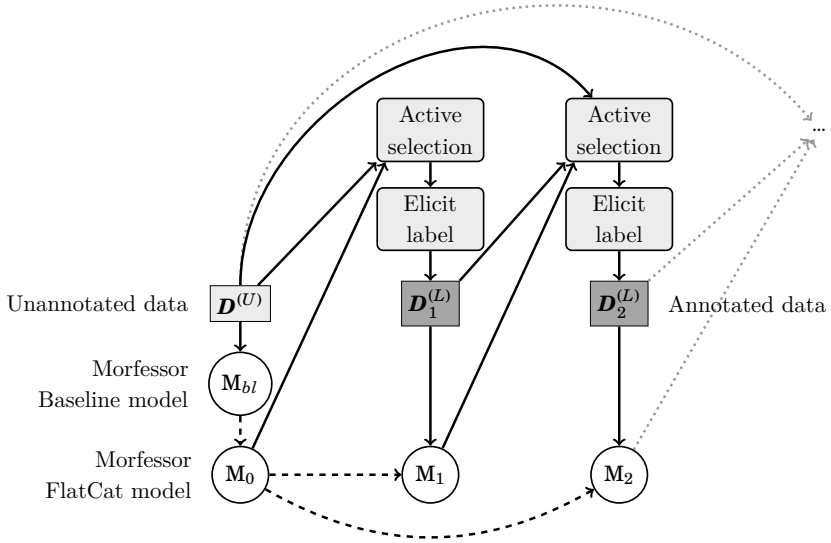


Figure 4.4. Active learning procedure for eliciting morphological segmentation annotations.

as input at time t the concatenation of the encoder output at time t and an embedding of the predicted label at $t-1$. The attention mechanism has been removed, with the encoder-to-decoder conditioning using a hard-coded index that always moves one step forward. Alternatively, the time-dependent connection to the encoder could also be described as a hard-coded diagonal monotonic attention. Note that the dependence of the decoder on previous decoder states and previous predictions is unmodified from the standard LSTM.

Decoding uses a lightly modified beam search, in which the probability of the end-of-sentence symbol is modified to ensure output of the correct length. Sorokin (2019) decodes from a convolutional sequence tagger using the Viterbi algorithm. While the Viterbi algorithm is better suited to the task, the required modifications to the software are also larger.

The model proposed in Publication IV is much smaller than the seq2seq baseline, requiring only 5% of the number of parameters. The reduction is due to the much smaller target vocabulary, and the fact that the proposed model requires no attention mechanism. The optimal network size in terms of number of layers and vector dimensions was also smaller in the experiments.

4.4.6 North Sámi morphological segmentation data set

Publication III presents a data set for semi-supervised training of morphological segmentation for the Uralic low-resource language North Sámi, collected using a novel active learning setup. The data set is based on two corpora: The list of unannotated words is computed from *Den samiske tekstbanken* (Sametinget, 2004), and the pool of words to annotate from the *UIT-SME-TTS* corpus. There

	Setup	Publication	$\|\boldsymbol{D}^{(L)}\|$	Pre ↑	Rec ↑	F_1↑
Morfessor EM+Prune	U	V	0	81.9	72.1	76.7
Morfessor Baseline	U	II	0	76.3	76.3	76.3
Morfessor Baseline	U	V	0	85.0	68.5	75.9
Adaptor Grammar	U	II	0	62.2	84.4	71.7
Morfessor FlatCat	U	I	0	84	60	70
SentencePiece	U	V	0	75.9	61.9	68.2
Morfessor CatMAP	U	I	0	89	51	65
CRF	SS	II	1000	89.3	87.0	88.1
CRF	S	II	1000	91.6	81.2	86.1
Morfessor FlatCat	SS	II	1000	86.9	85.2	86.0
Morfessor Baseline	SS	II	1000	84.4	83.9	84.1
Adaptor Grammar	SS	II	1000	76.7	82.3	79.4

Table 4.3. Boundary Precision (Pre), Recall (Rec), and F_1-score results for the *Morpho Challenge 2010* English test set. Methods in boldface are contributions of this thesis. Learning setups are abbreviated unsupervised (U), supervised (S), and semi-supervised (SS). $|\boldsymbol{D}^{(L)}|$ indicates the number of annotated samples used in training, and ↑indicates that higher scores are better.

are two published versions of the annotation data set.[24] The sizes of the data sets are shown in Table 4.2.

In the annotated part of the data set, each word is associated with one or more morphological segmentations. Each morph is additionally categorized as a prefix, stem, or suffix, e.g.

buokčevuojažiid buokče/STM vuoja/STM ži/SUF id/SUF

In addition to inflectional suffixes, also derivational suffixes that convert nouns into verbs are segmented, if the boundary is distinct. An exception was made in the case of certain lexicalized stems, appearing to end with a derivational suffix which has lost its conventional function. A further challenge was posed by the extensive stem alternation and fusion in Sámi. Consistency was maximized by placing the morphophonological alternation on the stem side of the segmentation boundary.

The annotations were produced by a single Sámi scholar, who is not a native speaker of Sámi. As a quality control, a second non-native Sámi speaking linguist independently reannotated 815 of the words, resulting in Cohen's kappa (Cohen, 1960) of 0.82, within the definition of *almost perfect* inter-annotator agreement. The annotations were collected using an annotation tool developed specifically for this purpose.[25] Words to be annotated were shown in context, to allow disambiguation of homonymous inflections.

[24]Available from `http://research.spa.aalto.fi/speech/data_release/north_saami_active_learning/`

[25]Software available at `https://github.com/Waino/morphsegannot`.

	Setup	Publication	$\lvert\boldsymbol{D}^{(L)}\rvert$	Pre ↑	Rec ↑	F_1↑
Adaptor Grammar	U	II	0	68.1	68.1	68.1
Morfessor EM+Prune	U	V	0	72.0	55.8	62.9
Morfessor CatMAP	U	I	0	76	51	61
Morfessor Baseline	U	V	0	62.3	58.2	60.2
SentencePiece	U	V	0	75.7	49.3	59.7
Morfessor Baseline	U	II	0	70.2	51.9	59.7
Morfessor FlatCat	U	I	0	66	52	58
CRF	SS	II	1000	89.3	87.9	88.6
CRF	S	II	1000	88.3	79.7	83.8
Morfessor FlatCat	SS	II	1000	81.6	80.2	80.9
Morfessor Baseline	SS	II	1000	76.0	78.0	77.0
Adaptor Grammar	SS	II	1000	69.7	77.6	73.4

Table 4.4. Boundary Precision (Pre), Recall (Rec), and F_1-score results for the *Morpho Challenge 2010* Finnish test set.

Active learning was applied to collect the annotations more efficiently, improving over random selection. A brief collection effort of this type results in a very small set of annotated words, which is still enough for a significant improvement when used in semi-supervised training. Figure 4.4 shows the active learning procedure.

(RQ1.3) With 300 words annotated with our active learning setup, the performance of Morfessor FlatCat in morph boundary F_1-score sees a relative improvement of 19% compared to unsupervised learning and 7.8% compared to random selection. The improvement over random selection was consistent over several sets of words with different morphological patterns. The largest benefit of the annotations was in the modeling of suffixation.

With the largest amount of annotations, the best query strategy for North Sámi combines uncertainty sampling with representative sampling (see Section 3.4.1). Strong performance is also achieved using a new proposed strategy based on coverage of word initial and final substrings, called IFSUBSTRINGS. It is inspired by the feature selection method called COVERAGE by Druck et al. (2009), which increases the ***coverage*** of the feature space by selecting features that are dissimilar from already chosen features. A set of binary features $\Omega(\boldsymbol{w})$ is defined to be substrings starting from the left edge (initial) or ending at the right edge (final) of the word $\boldsymbol{w}$.

In IFSUBSTRINGS, the next annotation is selected according to

$$A_{t+1} = \operatorname*{argmax}_{\boldsymbol{w}\in\mathscr{A}} \sum_{s\in\Omega(\boldsymbol{w})} \mathrm{I}\left(s \notin \Omega(A_j)\ \forall j \in \{1\ldots t\}\right) \frac{\mathrm{C}(s)}{N_{|s|}} \tag{4.20}$$

$$\Omega(\boldsymbol{w}) = \left\{\boldsymbol{w}_{0:k}, \boldsymbol{w}_{(|\boldsymbol{w}|-k):(|\boldsymbol{w}|)} \mid k \in 1\ldots 5\right\} \tag{4.21}$$

where I is the indicator function. The occurrence count $\mathrm{C}(s)$ is normalized by the

	Setup	Publication	$\|\boldsymbol{D}^{(L)}\|$	Pre ↑	Rec ↑	F_1↑
Adaptor Grammar	U	II	0	72.7	76.5	74.6
Morfessor EM+Prune	U	V	0	84.8	58.7	69.4
Morfessor Baseline	U	V	0	78.2	58.4	66.9
SentencePiece	U	V	0	75.2	60.0	66.8
Morfessor Baseline	U	II	0	67.9	65.8	66.8
Morfessor CatMAP	U	I	0	83	50	62
Morfessor FlatCat	U	I	0	88	38	53
CRF	SS	II	1000	89.3	92.0	90.7
CRF	S	II	1000	90.0	87.3	88.6
Morfessor FlatCat	SS	II	1000	84.9	92.2	88.4
Morfessor Baseline	SS	II	1000	85.1	89.4	87.2
Adaptor Grammar	SS	II	1000	77.0	90.9	83.4

Table 4.5. Boundary Precision (Pre), Recall (Rec), and F_1-score results for the *Morpho Challenge 2010* Turkish test set.

average occurrence count for substrings of the same length $\boldsymbol{N}_{|s|}$. In a simulated experiment using Finnish, IFSUBSTRINGS gave the best performance for all sizes of annotated training data.

4.4.7 Summary of intrinsic evaluation

Tables 4.3 to 4.5 show BPR results for three languages in the *Morpho Challenge 2010* data set Kurimo et al. (2010a,b): English, Finnish, and Turkish. The training sets contain ca 878k, 2.9M, and 617k word types, respectively.

Table 4.6 shows BPR results for North Sámi. Training data consists of 691k word types extracted from *Den samiske tekstbanken* corpus (Sametinget, 2004). The test set is 796 word types from version 2 of the data set collected in Publication III.

Note that the unsupervised results use development set based tuning of model hyper-parameters, which for Morfessor methods means α-tuning (Equation 4.12). For Adaptor Grammars, tuning entails selecting the specific form of the morphological grammar and inferring values for hyper-parameters. Morfessor CatMAP is the exception, as it does not support tuning.

Morfessor EM+Prune ranks high in the unsupervised setting, having the best F_1-score for English, and second best for the other languages. For Finnish and Turkish the top scoring system is Adaptor Grammar, which performs less well for English and extends poorly to semi-supervised training. Adaptor Grammars have, to the best of my knowledge, not been applied to North Sámi. As Publications III and IV focused on the semi-supervised setting for which Adaptor Grammars are not well suited, I did not select it as a baseline. Morfessor FlatCat is strong when used with semi-supervised training, but not as suited for unsupervised training.

	Setup	Publication	$\lvert\boldsymbol{D}^{(L)}\rvert$	Pre ↑	Rec ↑	F_1↑
Morfessor Baseline	U	V	0	75.7	60.7	67.4
Morfessor EM+Prune	U	V	0	73.0	62.1	67.1
SentencePiece	U	V	0	65.3	61.3	63.3
Morfessor FlatCat	SS	IV	200	78.2	77.6	77.9
Morfessor FlatCat	AL	III	297	77.2	84.2	80.5
CRF	SS	IV	1044	86.3	85.2	85.7
CRF	S	IV	1044	87.7	83.3	85.4
Neural Sequence Tagger	SS	IV	1044	84.3	85.6	84.9
Seq2seq	SS	IV	1044	87.7	80.2	83.7
Neural Sequence Tagger	S	IV	1044	83.3	83.9	83.6
Seq2seq	S	IV	1044	86.9	78.6	82.5
Morfessor FlatCat	SS	IV	1044	74.3	84.1	78.9

Table 4.6. Boundary Precision (Pre), Recall (Rec), and F_1-score results for North Sámi. Note that even models marked semi-supervised (SS) have benefited from active learning, as the collected annotations are partly the result of active learning procedures.

(RQ1.3) Approximately 1000 annotations are sufficient for training a strong fully supervised CRF model. Even with this generous amount of annotations, the best performance is reached with semi-supervised training when using Morfessor features in CRF. The importance of semi-supervised training increases when the number of annotations decreases. Semi-supervised CRF with Morfessor FlatCat derived features is the recommended way of applying Morfessor when aiming for linguistic segmentation and having access to small amounts of annotated data.

Table 4.7 summarizes the most important similarities and differences between the generative probabilistic subword segmentation methods described in this chapter.

			Morfessor variants					
	Greedy Unigram	SentencePiece	Baseline	CatMAP	**FlatCat**	**EM+Prune**	**ORM**	**Cognate**
Model	Unigram LM	Unigram LM	Unigram LM	HMM	HMM	Unigram LM	Restricted Unigram LM	Complex
Loss function	ML	MAP	MAP	MAP	MAP	MAP	MAP	MAP
Prior	–	DP	2-part MDL	Complex	2-part MDL	2-p MDL + DP	2-part MDL	Complex
Training algorithm	EM+Prune	EM+Prune	Local search	Local search	Local search	EM+Prune	Local search	Local search
Initialization	Seed lexicon	Seed lexicon	Words	Baseline	Baseline	Seed lexicon	Words + restrictions	Word pairs
EM variant	Lateen-EM once	EM	–	–	–	EM / Lateen-EM	–	–
Stopping criterion								
Fixed sequence	–	–	–	✓	✓	–	–	–
Cost change threshold	✓	–	✓	–	✓	✓	✓	✓
Target lexicon size	✓	✓	Approximate	–	–	✓	–	–
N-best decoding	–	✓	✓	–	✓	✓	✓	✓
Sampling decoding	–	✓	–	–	–	✓	–	–
Count dampening	–	–	✓	–	✓	✓	✓	✓
Semi-supervised	–	–	✓	–	✓	✓	✓	–
Requires pretokenization	✓	–	✓	✓	✓	✓	✓	✓
Reference implementation	C++	C++	Perl, Python	Perl	Python	Python	Python	Python
Publication	Varjokallio et al. (2013)	Kudo and Richardson (2018)	Creutz and Lagus (2002) Virpioja et al. (2013)	Creutz and Lagus (2005a)	Pub. I (2014)	Pub. V (2020)	Pub. VI (2016)	Pub. VIII (2018)
Thesis Section	4.3.4	4.3.4	4.3.5	4.3.5	4.4.1	4.4.2	4.4.3	4.4.4

Table 4.7. Comparison of generative probabilistic subword segmentation methods.

5. Machine Translation

> “Translation can occur without understanding, and understanding can occur without the possibility of translation. (Lakoff, 1987) ”

This chapter begins with a brief overview of the task of translation, and continues with important milestones in the development of automatic systems for the task of translation. The sections on early history of machine translation and statistical machine translation are based on Koehn (2009) and Poibeau (2017). The history of neural machine translation is based on Goodfellow et al. (2016). For background and history of neural methods in NLP, see Section 3.2.2. Finally, relevant subfields of machine translation are surveyed, before proceeding with the contributions of the thesis.

5.1 Translation

Translation, whether performed by humans or machines, is the task taking a text written in one natural language and producing another text, in a different natural language, that communicates the contents of the original text as faithfully as possible. There are many aspects of this faithfulness, involving semantics, tone, style, and structure. Ideally a translation should be both adequate and fluent. ***Adequacy*** means that the translation maintains the semantic content of the original. ***Fluency*** refers to the translation being good, grammatical language, and also maintaining an appropriate style and tone.

Good human translation requires in-depth understanding of the source text subject matter, a deep knowledge of the target language, and an adequate knowledge of the source language. The asymmetry in needed language competence is due to the fact that subtle mistakes of grammar need only to be avoided in the target language.

For machine translation, the holy grail of fully automatic high-quality translation (FAHQT), or human-equivalent publication-ready translation, is a high bar for current systems. While claims of parity with human translators have been made (Hassan et al., 2018), the claims have been criticized for using an evaluation methodology that is not appropriate for making such claims (Toral et al., 2018; Läubli et al., 2020). To assess the usefulness of machine translation, it is therefore necessary to ask how the translation will be used. Commonly three use cases are identified: assimilation, dissemination, and communication.

In ***assimilation*** or ***gisting***, a user initiates a fully automatic translation for content produced by someone else, in a language the user does not master. The translation is used to get a rough idea of the contents of the text. If the user determines, based on the gist, that publication quality is required, a separate translation process will be started. The quality demands are therefore low, but the speed should be fast and cost should be low. There is a large demand for this type of translation. Free services such as Google translate or Bing have pressed the expected cost to zero.

Dissemination requires publication-ready quality. The translation is typically requested by the producer of the content, which enables quality-increasing approaches that are not available in the gisting use case. The required level of quality can be achieved either by controlling the input to the translation system using methods such as controlled language or pre-editing, or alternatively by having a human translator post-edit the output. Speed and cost are less constraining. The quality of the machine translation needs to be high enough to ease the work of the translator. If the quality is too low, it will be more efficient to translate without help from MT. It is also important to consider the way in which the machine translation is presented to the translator in the user interface.

In the use case of ***communication***, two or more users wish to communicate using different languages. Speed is of the essence: optimal translation would be simultaneous to the original speech. The quality does not need to be as high as for dissemination. Users may tolerate some errors, and it is possible to iterate by asking for clarification if the message is not understood.

5.1.1 Challenges

There are many challenges that make achieving adequacy and fluency difficult. Text is a form of communication between an author and one or more recipients. Translation is intended to adapt the communication for new recipients, who may not share the cultural background and world knowledge of the original recipients. The translator may need to paraphrase or explain certain expressions, if there is no direct correspondence. Finding an appropriate reformulation is difficult, especially for idiomatic and formulaic expressions. Explicating often requires adding external knowledge that is needed by the new recipient but not included in the source text. Natural language is inherently ***ambiguous*** in many ways. Attempting to resolve the ambiguity results in a circular problem where the meaning of words is determined from the context, but the meaning of the context is constructed from the individual words that it consists of.

Metaphorical language, idioms, and formulaic expressions pose a major challenge. Metaphorical language is surprisingly common, even outside of poetry and artistic prose (Lakoff and Johnson, 1980; Lakoff, 1987). The commonly used metaphors are not always shared between languages, necessitating the translator to devise a different metaphor capturing the same meaning, or to replace the metaphor with a more concrete expression. For example the Finnish expression

"heittää lusikka nurkkaan" should not be translated literally as *"throw the spoon in the corner"*, but instead either using a different idiom *"kick the bucket"* or non-metaphorically as *"die"*.

Current machine translation systems are only able to memorize metaphorical language from large data sets, but are not able to learn to generalize such rich and abstract concepts for analyzing language, or to combine them with "common sense" world knowledge. For this reason machine translation is at the moment best suited for factual domains, such as news text or technical text. Metaphorical language is used even in these domains, but to a lesser extent than in prose and poetry. Some examples from the domain of financial news include using the expression *"burning money"* when the money is merely being spent, or *"taking the temperature of the market"*. Scientific text benefits from being low on ambiguity and metaphorical language, but is instead challenging due to the importance of rare jargon terms and precise distinctions. In scientific and technical texts, the exact choice of term can be important, and fuzzy-matching to a related term may introduce subtle errors. The most difficult domains to translate are the ones that require sensitivity to cultural and artistic values. Some progress is being made even in these difficult domains. For example Toral and Way (2018) report that 17%–34% of machine translated sentences from three novels are perceived by native speakers to be of human-equivalent quality. Still, human translators of literature and poetry will not be made obsolete anytime soon.

Even mundane texts present a wide array of challenges. Language ***diversity*** is one such challenge. While closely related languages include many concepts, expressions, and linguistic attributes that can be mapped using a quite shallow analysis, the more distant the languages are, the more abstraction is required.

One result of language diversity is lexical asymmetry, e.g. when translating Finnish *"katto"* requires choosing between *"roof"* and *"ceiling"* as that distinction is not made in the source language.[1] Another type is syntactic asymmetry, e.g. whether certain properties are expressed through having (*"The poor don't have any money"*) or being (*"Köyhät ovat rahattomia"*[2]). There are various forms of morphological asymmetry, e.g. different gender and number systems. Sometimes there are even morphological long-range dependencies, such as when German verb prefixes are moved to the end of the sentence: *"Ich fahre morgen los"*.

As productive morphology results in a large number of word forms, morphologically rich languages suffer from ***data sparsity*** issues when estimating word-level statistics. Differing word order between source and target languages poses challenges for machine translation, as the units that compose the source must be reordered in addition to being translated. Free word order on the target side poses potentially an even larger problem, as statistics of discrete representations, e.g. n-grams, are further spread out over variations in word order. The translation system needs to both select a reasonable word order, and make sure that morphological markings end up on the correct target words. Even evaluation

[1] See discussion on *valeur* in Section 2.1.

[2] Finnish. Literal translation: the poor are without money

is more challenging for morphologically rich languages, as standard evaluation measures operate on word-level statistics.[3]

The data can vary in quantity, quality, and appropriateness of domain. When using machine learning, the domain or domains from which the training data are gathered affects the usable vocabulary. To correctly use the jargon, or domain specific terminology, requires training data from that domain. Even a large amount of data from a different domain might not be helpful in learning those terms. Typically all three challenges affect the low-resource languages: when data is hard to come by, even noisy and out-of-domain data must be used.

The challenges of ambiguity, data sparsity, and difficult evaluation are affected by subword segmentation. Translational ambiguity can be increased by target side morphological complexity, when one word can align to multiple different inflections. Segmenting the stem from the affixes removes this extra ambiguity, freeing up more resources for modeling the irreducible translational ambiguity. A poor segmentation might increase ambiguity, by introducing arbitrary distinctions through inconsistent segmentation decisions. Segmentation alleviates data sparsity, but at the cost of longer symbol sequences. Long sequences cause longer dependency spans, and may require larger computational resources. Evaluation can be made more reliable by considering subword information.

For some sentences the challenges are temporarily absent, and the shallow understanding of current methods is enough to produce a perfect translation. Only viewing such happy occasions can give a sense that the problem of translation is close to being solved, but perhaps this is an Eliza-effect.[4] To avoid anthropomorphizing machine learning systems, it is important to carefully analyze the errors they make.

5.1.2 Reducing other tasks to machine translation

When speaking of applications of machine translation, it is useful to separate actual translation tasks from tasks that resemble translation and can be solved using translation software. Translation always involves mapping between two or more natural languages. The natural language may be represented in various modalities, e.g. text, speech, or sign language. It is possible that source and target are in different modalities. The language can be used for communicating knowledge from various domains, e.g. news, biomedical, or social media. We can therefore define subtasks, such as text-to-text news translation, or speech-to-speech meeting translation.

Related, translation-like tasks involve mapping from one sequence to another so that one or both sequences comprise natural language. These tasks include summarization (Rush et al., 2015), parsing (Dyer et al., 2015), question answering (Yin et al., 2016), generating interpretive text (e.g. image captions Bernardi et al., 2016), and morphological tasks (Kann et al., 2018). In cutting edge re-

[3]Evaluation of MT is discussed further in Section 5.4.

[4]See Chapter 1 on the ELIZA chatbot.

search, an attempt to translate brain activity to text has been made (Makin et al., 2020). The methods can also be applied to non-linguistic string manipulation systems, such as DNA sequences, music score sheets, or transcripts of chess games.

5.1.3 Approaches

One of the important decisions when building a machine translation system is the choice of translation units, i.e. how large chunks of text to translate at a time. In machine translation, the main reason for restricting the translation unit is computational, but even human translators divide a text in order to make it more manageable and conserve working memory. However, human translators are able to retain context from previously translated units. ***Translation equivalents*** are cross-lingual near-synonyms, either words or phrases. Most sentences are not fully decomposable into translation equivalents, unless the languages are very similar.

Word-for-word translation is an approach that results in low quality translations, except perhaps in rare cases involving closely related languages. Word-for-word translation produces stilted language, where the structures of the original language shine through in unnatural ways. In the worst cases the result is a "word salad" that is very difficult to understand. Good translation depends on analogies between concepts rather than a simple mapping of words and expressions. Current machine translation systems do not perform word-for-word translation. However, when moving to larger units, a similar problem recurs. Most systems decode a sentence at a time and do not retain context between sentences. Statistical machine translation systems are not even truly sentence level, as they construct their hypothesis from shorter ***statistical phrases***. The idea of ***constructionism*** is to divide text into relatively independent units from which sentences can be constructed. Current state-of-the art neural machine translation systems are able to use the full ***sentence context***, although they can still be fooled by a misleading strong local context. ***Document-level*** or discourse level information is only rarely used by current systems. Even systems described as document-level typically only use neighboring sentences as context (Tiedemann and Scherrer, 2017; Popel et al., 2019). Some exceptions using full document context exist (Hardmeier et al., 2013; Junczys-Dowmunt, 2019).

Systems can be divided based on the ***level of abstraction*** into three categories: direct, transfer, and interlingual systems. The Vauquois triangle, reproduced in Figure 5.1, visually represents the level of abstraction of the transfer. ***Direct translation*** systems map the source directly into the target, which originally meant a word-for-word translation. ***Transfer translation*** involves performing morphological, syntactic, and/or semantic analysis of the source, mapping the resulting structures using transfer rules, and finally generating the target language by generation steps that are the inverse of the analysis steps. Transfer is generally fit to a particular language pair. ***Interlingual*** systems extend the analysis all

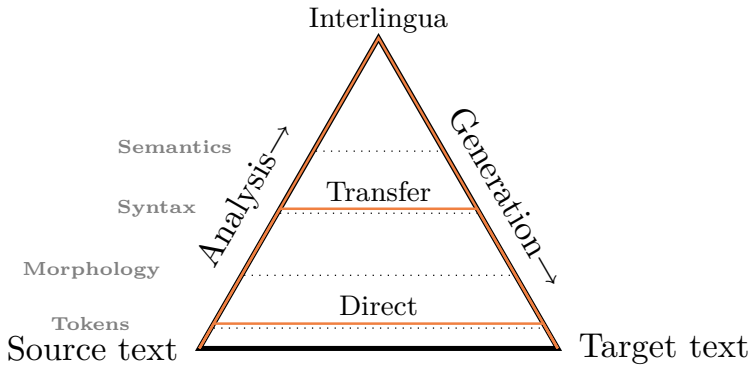


Figure 5.1. Vauquois triangle, showing machine translation as a combination of analysis raising the level of abstraction, and its inverse, generation. Although only syntactic transfer is shown, transfer can occur at any level.

the way to an interlingua, a language-independent conceptual form, where all ambiguity is resolved. Performing the analysis requires introducing external world knowledge that is not explicit in the document. The amount of information that must be added to the interlingual representation by inference from contextual and world knowledge has no clear limit.

Processing in the form of analysis and generation cannot truly add information to the text. It can transform and rearrange the information into a more accessible form, e.g. by inferring information from the global structure of a sentence and making it available from local context by adding some annotations. It can also destroy information e.g. when misspellings or casing variants are normalized. In practice, an attempt is made to turn implicit information into explicit. This involves guessing, and the risk of incorrect analysis.

Systems are further divided according to the machine learning paradigm: rule-based (RBMT), example-based (EBMT), statistical (SMT), and neural (NMT) machine translation. The paradigms are discussed in the upcoming sections.

All the data-driven paradigms, i.e. all the paradigms except for rule-based machine translation, share the property that they define conditional language models $P(t_j \mid t_{0:(j-1)}, \boldsymbol{s})$, which generate the target sequence $\boldsymbol{t}$ conditioned on both previously generated target tokens and the source $\boldsymbol{s}$.

5.2 From historical to modern machine translation

This section presents a brief summary of the history of machine translation.

5.2.1 Early machine translation

The idea of automatic translation was invented before there was any way to practically implement such a system. To the best of my knowledge, the first time the idea was formalized was in a 1939 patent by Petr Petrovich Troyanskii (Hutchins and Lovtskii, 2000). After the second world war, electronic computers developed

at a rapid pace, and began to be applied also to NLP tasks. There was both a practical need for machine translation, e.g. to translate texts from Russian to English for the use of Cold War military intelligence, but also a scientific interest in empirical exploration of how language works.

The first machine translation system to be built was the demonstration system of Georgetown University in 1954. It used a bilingual dictionary of 250 words and 6 grammar rules to translate a cherry-picked set of 49 Russian sentences into English. From the 1940s to the 1960s, several teams in the USA, Great Britain, USSR, and other countries worked on computer-based translation systems. This was a period of optimism, as the extent and difficulty of the challenges involved in machine translation were not yet known.

The period of optimism ended with the Bar-Hillel (1959) report and the famous ALPAC report (Automatic Language Processing Advisory Committee, 1966). Even though the reports recommended more research into fundamental computational linguistics and computer aided human translation (CAT), the overall tone of the reports was very negative. The reports questioned both the practicality of implementing machine translation of sufficient quality, and also the volume of demand for translation compared to the number of available human translators. As a result, the reports were followed by a "machine translation winter". Funding was reduced in the US, and enthusiasm waned.

Despite the reduced research interest, the first commercial systems were developed in the following decade. The company SYSTRAN was founded in 1968. The system was initially rule-based direct translation, but would later develop into a hybrid system. TAUM-Météo, a rule-based system for translating weather forecasts, started operating in 1976. It was developed at the University of Montreal. The focus on a domain with restricted vocabulary and limited syntax made the rule-based translation approach very successful.

5.2.2 Data-driven machine translation

A new period of rapid advances started in the late 1980s. It was enabled by two prerequisites: the collection of sufficiently large corpora of parallel text, and development of computers with sufficient capacity to process them.

The two main paradigms of machine translation of this era are ***example based machine translation*** (EBMT) and ***statistical machine translation*** (SMT). Transfer rules are complicated to write for a language pair such as English-Japanese, with very different structures and fixed word order. EBMT (Nagao, 1984), obviates the need for explicit restructuring rules by using analogy from examples. The process bears a resemblance to how human translators use translation memory: Software is able to retrieve examples of how similar fragments were translated, to aid the translator with suggestions. EBMT is a nonparametric method that stores the training corpus in raw form. During translation, the most relevant training sentence pairs are retrieved, and a translation is constructed as a synthesis of fragments extracted from them. Relevant sentences are such that

contain fragments of the sentence to be translated.

The ***noisy channel*** model of communication from information theory by Claude Shannon and Warren Weaver (Shannon, 1948) found many practical uses e.g. in telecommunications, cryptography, and speech technology. It also acted as an inspiration to statistical methods in machine translation.

> “ When I look at an article in Russian, I say:
> ‘This is really written in English, but it has been coded in some strange symbols.
> I will now proceed to decode’ (Weaver, 1955) ”

The probability of the target sentence given the source sentence $P(\boldsymbol{t}\,|\,\boldsymbol{s})$ is decomposed using Bayes' theorem into the probability of $\boldsymbol{s}$ emerging from the noisy channel when the input is $\boldsymbol{t}$, called the translation model (TM), and the language model (LM) probability of the hypothesis target sentence

$$P(\boldsymbol{t}\,|\,\boldsymbol{s}) = \overbrace{P(\boldsymbol{s}\,|\,\boldsymbol{t})}^{\text{TM}}\,\overbrace{P(\boldsymbol{t})}^{\text{LM}}. \tag{5.1}$$

SMT is based on statistics computed from the training corpus, e.g. language internal collocations such as n-gram frequencies, and cross-lingual collocations such as alignment frequencies. A high-probability translation is decoded using these statistics. This factorization comes with several practical benefits. Firstly, it allows a simpler distribution to be used for the translation model. A word-for-word translation model can rely on an n-gram language model to account for target language word order. Secondly, the language model can be trained from monolingual data, which is typically available in much larger quantities than the parallel training data.

Seminal work from the IBM research group includes the IBM alignment models 1 to 6 (Brown et al., 1990, 1993) and the Candide translation system (Berger et al., 1994). The IBM word alignment models are still in use at the time of writing. The IBM models incrementally add complexity, with (1) lexical, (2) absolute position, (3) fertility, and (4) relative position -based alignment. IBM 5 corrects for deficiency, and IBM 6 adds an HMM alignment model.

All translation systems can be viewed in the analysis-transfer-generation framework presented in Figure 5.1. In early word-level SMT, analysis and generation had become vestigial, with only simple tokenization and case normalization. A challenge is presented by the need for the generation component to be invertible, so that training pairs for the transfer component can be produced from parallel data in surface form.

Phrase-based statistical machine translation. While IBM models allow one-to-many alignments using fertility, they can not model many-to-many alignments. This is problematic for expressions requiring rephrasing and in particular idiomatic expressions. Automatic phrase extraction begins from word alignments. Many-to-many alignments can be acquired by symmetrizing a source-to-target

and target-to-source alignment, e.g. with the commonly used heuristic GROW-DIAG-FINAL-AND (Koehn et al., 2003). Contiguous aligned sequences (statistical phrases) can then be extracted. Limiting extraction of phrases to spans that cover a linguistic constituent is detrimental.

In phrase-based statistical machine translation (PB-SMT), a distinction can be made between the tokens and the translation units. The sentences are tokenized into lexical units (typically words, but also subwords have been used), but the translation hypothesis is built by concatenating longer sequences of tokens.

PB-SMT systems combine several submodels (feature functions f_k) with a weighted log-linear model

$$P(\boldsymbol{t},\boldsymbol{a}\,|\,\boldsymbol{s}) = \frac{1}{Z(\boldsymbol{s})}\exp\left(\sum_{k=1}^{K}\lambda_k f_k(\boldsymbol{s},\boldsymbol{t},\boldsymbol{a})\right), \tag{5.2}$$

where $\boldsymbol{a}$ is the alignment, $\boldsymbol{Z(s)}$ the partition function, and λ_k the submodel weights. The most prominent translation system of this time is the PB-SMT system Moses (Koehn et al., 2007).

PB-SMT can be seen as taking a small step in the direction of EBMT: the output is composed out of literal examples in the form of entire extracted phrases, and the choice of examples can be based on incorporating linguistic data e.g. using a class-based LM or factored SMT (Koehn and Hoang, 2007).

5.2.3 Neural machine translation

The millennium shift saw a rapid increase in the amount of digital corpora from the Internet, which propelled the performance of SMT systems to new levels before ultimately reaching a plateau. The large data sets also fueled a paradigm shift to neural machine translation (NMT) in the early 2010s, inspired by the successes of deep learning in other fields.[5]

However, this was not the first time that neural methods were applied to NLP or even MT. One early work by Chrisman (1991) introduces the neural ***encoder-decoder*** architecture to MT. The neural network is divided into two subnetworks: the ***encoder***, which reads in input and encodes it into an intermediate representation, and the ***decoder***, which generates the output from the intermediate representation. It does not yet use the modern recurrent neural network conditional language model decoder, instead basing the encoder and decoder on sequential recursive auto-associative memory (S-RAAM). The method requires an invertible encoding, in which both source and target are independently encoded into the same interlingual representation, making the model unable to handle translational ambiguity.

The early NMT systems were not usable in practice. They were tested in experiments with very small vocabularies and short, controlled sentences. The first generally applicable systems used neural components as part of SMT systems. Instead of replacing the entire translation system with a single neural network,

[5] See Section 3.2.2 for a general introduction to deep learning.

the overall modular structure of the SMT system was retained, and individual modules were replaced or augmented with neural networks. Schwenk et al. (2006) use a neural network language model (NNLM) instead of an ***n***-gram language model. Devlin et al. (2014) add conditioning on the most likely source words based on the alignment information from the decoder to the NNLM. Son et al. (2012) use a neural network as the translation model. This allows using larger context without suffering from sparsity-induced estimation problems. Cho et al. (2014b) improve the estimated translation probabilities in the phrase table of a PB-SMT system, by rescoring with an encoder-decoder architecture based on the new GRU.[6]

Inspiration for end-to-end NMT architectures came from success in automatic speech recognition (Graves et al., 2013). Seminal works by Kalchbrenner and Blunsom (2013) and Sutskever et al. (2014) approach parity with state-of-the-art SMT systems. Both use encoder-decoder architectures, with recurrent conditional language model decoders. The encoder of Kalchbrenner and Blunsom (2013) is convolutional, while Sutskever et al. (2014) uses a deep LSTM recurrent architecture instead. The convolutional network is not sensitive to the global order of words in the sentence.

A bigger flaw shared by both methods is the ***bottleneck***, which encodes the entire source sentence into a single vector. Sutskever et al. (2014) try to alleviate the resulting long dependency chains by reversing the order of the words in the source sentence. This shortens the distance between the beginnings of source and target, making it easier for the decoder to start generating the correct translation. Ensuring that the end of the sentence is generated correctly must rely more on target language modeling, as the distance from the source grows rapidly. The average distance between source and target words is unchanged. Cho et al. (2014a) analyze the properties of encoder-decoder architectures with fixed length representation. They find that this type of architecture works well on short sentences, but performance rapidly decays as the length of the sentence or the number of unknown words increases. The single vector bottleneck was not problematic for Cho et al. (2014b) as the encoder only needs to represent a phrase rather than a complete sentence. When used in end-to-end neural machine translation, the bottleneck becomes a problem.

The attention mechanism[7], applied to MT by Bahdanau et al. (2014), provides a more complete solution to the problem, by creating a controllable connection from the decoder directly to all timesteps of the encoder. In addition to allowing the decoder to decide which encoder timesteps are most relevant to the symbol currently being produced, attention also shortens backpropagation paths, and increases the flow of information from encoder to decoder.

Although the attention mechanism was first used in combination with a recurrent encoder and decoder, Vaswani et al. (2017) show with their Transformer architecture that the attention mechanism is sufficient for sequence modeling, ob-

[6]Recurrent units presented in Section 3.2.2.

[7]The attention mechanism presented in Section 3.2.2.

viating the need to use recurrence or convolution. Transformers currently hold the state-of-the-art for many NLP tasks.

A Transformer is a deep stack of layers, consisting of two types of sub-layer: multi-head attention sub-layers and feed-forward sub-layers. Each sub-layer is individually wrapped in a residual connection (He et al., 2016b) and layer normalization (Ba et al., 2016). A layer normalized unit recenters and rescales its activations using normalization terms computed separately for each example. When used in translation, Transformer layers are stacked into an encoder-decoder structure. In the encoder, the layer consists of a self-attention sub-layer followed by a feed-forward sub-layer. In the self-attention, the output of the previous layer is used as queries, keys, and values $\boldsymbol{Q} = \boldsymbol{K} = \boldsymbol{V}$. The decoder includes a third sub-layer, the context attention, which is inserted between the self-attention and the feed-forward sub-layer. In context attention, $\boldsymbol{Q}$ is again the output of the previous layer, but $\boldsymbol{K} = \boldsymbol{V}$ is the output of the encoder stack.

5.3 Subfields of machine translation

This section surveys some subfields of machine translation that are relevant to the topic of the thesis: how to represent the vocabulary items for MT from and into morphologically rich languages, MT in low-resource settings, the use of cross-lingual transfer through multilingual MT, and ways of exploiting monolingual data.

In order to train a MT system for a language pair SRC→TRG, three distinct types of resources may be useful: (i) ***parallel*** data for the language pair of interest SRC–TRG, (ii) ***monolingual*** data in either SRC or TRG, and (ii) data in ***related*** languages, which can be either parallel data for other language pairs (SRC–X, X–TRG), or monolingual data. In the low-resource translation setting, it is primarily the parallel data that is scarce. Monolingual data is easier to acquire and typically more abundant. In addition, there may be related languages with much more abundant resources.

5.3.1 Vocabulary construction

First, methods for constructing the vocabulary for the machine translation system are considered. Some related ways of incorporating morphological information into MT are also included. As the motivation for the use of subword segmentation in NLP was given already in Chapter 4, this Section focuses specifically on the translation setting. For a review of subword segmentation methods, see Section 4.3.

Subword segmentation has become a standard in NMT. For example, in WMT 2019 shared tasks, two thirds of submissions used byte pair encoding (BPE), and a quarter of submissions used SentencePiece (Barrault et al., 2019). The use of subword segmentation is no longer limited to morphologically rich languages, and

it has seen use in state-of-the-art methods in high-resource settings (Arivazhagan et al., 2019b; Ng et al., 2019; Junczys-Dowmunt, 2019).

The problem of rare words

Word-level vocabularies may grow to unmanageable size, particularly in morphologically rich languages.[8] In addition, it may be necessary to restrict the vocabulary of an NLP system for technical reasons, e.g. due to the difficulty of estimating needed statistics for low-frequency items. The model capacity or computational constraints might also be limiting factors.

Using a frequency cutoff is a simple way to perform the vocabulary restriction. Words with a frequency below a chosen cutoff are treated in some special way. Based on the frequency cutoff, the total vocabulary can be divided into three parts: the ***frequent words*** included in the truncated vocabulary, the ***rare words*** which are present in the training corpus but not in the truncated vocabulary, and the ***out-of-vocabulary*** (OOV) words that are not included in the training corpus at all. Rare words are a challenge for machine translation, and due to the long tail of the Zipfian distribution, there is a large number of them.

Copyable words are a special class of rare words, for which there is a regular mapping between the surface forms in the source and target. Due to the arbitrariness of the sign[9], this is not usually the case. In the prototypical copyable word, the source and target are identical character-for-character. In other cases some transliteration is needed, in particular when the languages differ in orthography. Proper names and some loan words are prominent examples of words in this class. In some morphologically rich languages, proper names are inflected, e.g. Finnish *"Münchenistä Bordeaux'hon"* (From Munich to Bordeaux). A part of the string is copyable even in the case of inflected names. Cognates and loanwords can undergo regular phonological and morphological transformations, e.g. Latin *"claustrophobia"* and German *"Klaustrophobie"*.

When the size of the vocabulary is restricted, the question of how to represent the words outside the vocabulary arises. Different tasks have different requirements for the representation of OOV words. In natural language understanding tasks, e.g. classification, it may not be necessary to retain all of the information contained in rare words. Collapsing non-discriminative distinctions—e.g. misspellings, case variants, or some inflections—may be desirable. In natural language generation tasks, on the other hand, the representation must include everything that is needed to generate the correct surface form. Machine translation combines natural language understanding in the encoder with natural language generation in the decoder. In a typical MT setup, there are 2–4 possible locations where a representation is needed, as shown in Figure 5.2. It is possible to use different representations in each location.

The simplest approach discards any rare and OOV words not included in the truncated vocabulary. A slightly better solution instead substitutes a spe-

[8] See Section 2.1 for a discussion of vocabulary growth.

[9] See Section 2.1 for the arbitrariness of the sign.

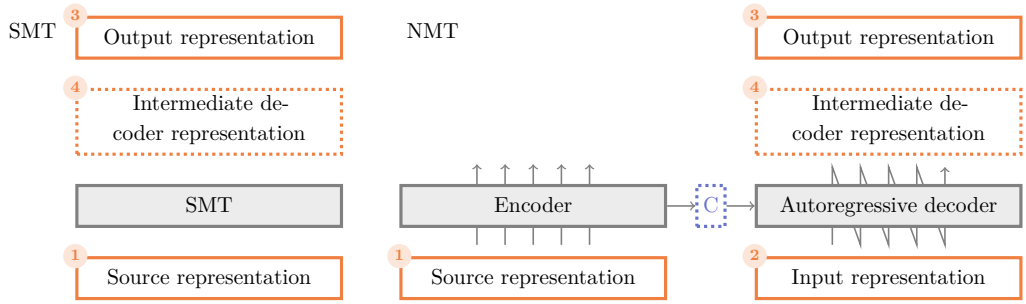


Figure 5.2. Four locations in MT where representations may need to be specified. Due to the autoregressive nature of the NMT decoder, it also needs an input representation 2. The intermediate decoder representation 4 is optional. The continuous intermediate representation C is learned, not specified.

cial ⟨UNK⟩[10] token. Schwenk (2007) restricts the output vocabulary of a neural language model to a short-list composed of the most frequent words. This approach is adequate when rare words can be ignored without much loss, e.g. in understanding-oriented tasks, or when the neural language model is combined with other models that use larger vocabularies. However, if the neural model is used for generating natural language on its own, the ⟨UNK⟩ approach is not appropriate as output representation 3. Some alternate method that allows the system to generate also the rare words is needed.

One approach aims to alleviate the problem by enabling the use of very large vocabularies. Representations 1 and 2 are easy to expand, but the output distribution for 3 is problematic. Stochastic approximations of the categorical distribution e.g. using noise contrastive estimation (Gutmann and Hyvärinen, 2010; Jean et al., 2015) can be used. Alternatively the draw from the categorical distribution can be decomposed into a sequence of smaller draws, e.g. with hierarchical softmax (Morin and Bengio, 2005) or structured output layers (Le et al., 2011). Self-normalization (Devlin et al., 2014) avoids the heavy computation of the normalization denominator by learning weights that result in an approximately normalized distribution.

The methods mentioned above are not specific to translation, and can be applied to a wide range of suitable NLP models. In translation-specific approaches, the class of copyable words can be approached with some methods particular to this class. The ***copy mechanism*** (Luong et al., 2015b) uses numbered ⟨UNK⟩s to delegate handling of rare words to a post-processing step. The ⟨UNK⟩s in the source are numbered consecutively, and the decoder can refer to them using the number. In a post-processing step, the referred words are either translated using a dictionary, or copied without modification into the target sentence. For example, translating from Swedish to English,

[10]The usage of the OOV and ⟨UNK⟩ terms follows established practice, although it is not consistent with using the term ***vocabulary*** for the fixed-size model vocabulary.

Xyzzy är det magiska ordet i ADVENT ↦
⟨UNK$_1$⟩ är det magiska ordet i ⟨UNK$_2$⟩ ↦
The magic word in ⟨UNK$_2$⟩ is ⟨UNK$_1$⟩ ↦
The magic word in ADVENT is xyzzy .

If a copying mechanism is available, rare words are best kept together as a single unit, to avoid copying errors. A copying mechanism is unable to inflect the copied words. For such languages, the subword segmentation approach may be preferable. When using subword segmentation, it is instead beneficial to segment rare words heavily. If the segmentation is also consistent, it becomes possible to decode the copyable words piece by piece, with the decoder attending to each copyable piece in turn.

English-centric models

It has long been common in machine translation research to select English as the target language. When English is not the target, it is typically the source. Even experiments in massively multilingual settings are typically ***English-centric***, using English as target, source, or both in turn (Arivazhagan et al., 2019b; Mueller et al., 2020). When a morphologically rich language (MRL) is present, it is typically on the source side with English as the target. There is a recent thesis by Passban (2018) focusing on translation *from* MRLs. Ongoing trends, e.g. recent WMT shared task campaigns, are slowly bringing a change to the prominence of English. Some exceptions involving e.g. translation between related languages (Zhang, 1998; Tiedemann, 2009; Costa-jussà et al., 2018) and multilingual NMT with experiments where neither side is English (Vázquez et al., 2019; Platanios et al., 2018) exist. In spoken language translation, Bansal et al. (2019) translate from the extremely low-resource Mboshi to French. Virpioja et al. (2007) use a MRL on both source and target side, and both sides are segmented into subwords.

The results of Koehn (2005) shed light on the difficulty of translating into MRLs from typologically dissimilar source languages. They train 110 pairwise SMT systems to translate between all of the languages in the Europarl dataset. Finnish has the lowest average quality both as a source language and as a target language. German and English are equally difficult to translate from, but German is much more difficult when used as target language. While NMT has to some extent leveled the playing field, results of WMT news translation shared tasks 2018 and 2019[11] give some rough indication that Finnish remains a difficult target language. BLEU scores for English→Finnish have remained much worse (8 BLEU or more) than higher-resourced languages such as Russian, German, and Chinese. Also, Finnish→English BLEU scores are higher (5.6–6.6 BLEU) than English→Finnish scores, while for Russian, German, and Chinese the asymmetry is smaller or

[11]Even though English↔Finnish was included as a language pair also in 2017, the year is excluded from these observations as the transition to the NMT paradigm was still in progress.

even in the opposite direction.[12] When extending the analysis to the online reference systems, the asymmetry in favor of Finnish→English increases, with large differences of 7.8–9.4 BLEU.

As a counterargument, word-level metrics such as BLEU are less well suited to morphologically rich languages,[13] which may result in overall lower scores in the direction into MRLs. However, as Russian and German are morphologically rich, these discrepancies point at other contributing factors, such as the amount of training data. Using additional metrics that are more suited to MRLs shows that the choice of metric partly explains the asymmetry. Measured using LEBLEU,[14] the difficulty of Finnish as target language is confirmed when comparing to German but not when comparing to Russian. Finnish→English is 3.9–6.0 LEBLEU better than the reverse English→Finnish. For German, the opposite is true: English→German is better by 0.9–1.1 LEBLEU compared to the reverse. Russian→English is is better by 3.0–7.4 LEBLEU compared to the reverse, which is close to the Finnish asymmetry. Measured using CHRF-2.0, results are inconclusive.

Morphology in machine translation

Morphology has played an important part in machine translation since the beginning. Already the early rule-based transfer systems included morphological analysis of the source and morphological generation on the target side.

The introduction of SMT, with the word-level model of Brown et al. (1990), represented a momentary return to direct translation. It did not take long for Brown et al. (1992) to propose an extension into a statistical transfer system, by adding pre- and postprocessing. The authors argue that all systems require at least some kind of preprocessing, and thus there exist no systems purely in the class of direct translation. Concerning basic preprocessing they discuss the challenges of tokenization, and introduce true-casing. A morpho-syntactic analysis is performed on both source and target. They annotate words according to grammatical function (part-of-speech tag), perform some syntactic reordering, extract inflectional morphology and perform some word sense disambiguation. Since then these useful techniques have been reintroduced and extended upon several times.

In SMT, it is possible to improve individual component models—such as the alignment model (Popović and Ney, 2004) or the language model (Botha and Blunsom, 2014)—by making them morphology-aware.

Morphological generation and prediction are two related approaches to dealing with morphologically rich target languages. ***Morphological generation*** requires a fully specified morphological intermediate representation 4, typically implemented as a factored output representation (Koehn and Hoang, 2007; Avramidis

[12]Higher scores when translating *from* English, together with a larger number of submissions in that direction, may be seen as an indication of increased interest in non-English target languages.

[13]Evaluation metrics discussed further in Section 5.4.

[14]For LEBLEU, a contribution of this thesis, see Section 5.5.1.

and Koehn, 2008; García-Martínez et al., 2016, 2020). Factored translation works well with morphologically moderate languages such as French (García-Martínez et al., 2020). Yeniterzi and Oflazer (2010) map English syntactic features into factors needed to generate Turkish morphology. Factored input can also be used on the source side 1 (Sennrich and Haddow, 2016). ***Morphological prediction*** uses a multi-step translation process with an underspecified intermediate representation 4, e.g. a sequence of stems (Toutanova et al., 2008; Clifton and Sarkar, 2011). Talbot and Osborne (2006) learn what to simplify based on minimizing lexical redundancy. Passban et al. (2018) augment a character-level decoder with a second attention mechanism to retrieve information from an external morphology table.

In a ***factored*** representation, each word is represented by a tuple of factors, e.g. lemma and morphological tags. An alternative is to ***interleave*** the different types of information into a single sequence. Already Brown et al. (1992) use an interleaved representation, containing words, morphs, abstract tags, and tokens joining several of these:

> He was eating the peas more quickly than I. ↦
>
> He PAST_PROGR to_eat the pea N_PLURAL quick er_ADV than I.

More recent applications of interleaved intermediary representations include Goldwater and McClosky (2005) and Tamchyna et al. (2017).

Language divergence is a significant challenge in MT. Many efforts have focused on ***increasing the symmetry*** between languages in order to improve alignment. Asymmetric granularity can be caused e.g. by compounding, when one language uses a closed compound (*"sähköpostijärjestelmä"*) to represent a concept that is written as an open compound in another language (*"e-mail system"*). Asymmetric marking of grammatical properties can cause ambiguity, such as when genderless *"hän"* aligns to both *"he"* and *"she"*.

Splitting closed compounds (Brown, 2002; Koehn and Knight, 2003; Popović et al., 2006; Fritzinger and Fraser, 2010; Stymne and Cancedda, 2011; Huck et al., 2017) is a simple yet effective way to increase symmetry. Compound words can sometimes be translated compositionally, by splitting the compound into parts, translating each part individually, and possibly rejoining the translated parts into a new compound.

In ***segmented translation***, the words are split into subwords. Work in segmented translation has often started with a linguistically motivated morphological analysis, yielded e.g. by a handcrafted analyzer implemented as a finite state transducer (FST). Language-specific heuristics are used for selecting which parts of the linguistic analysis should be represented. A typical approach is to select a highly inflecting class of words and segment them for particular morphological features (Nießen and Ney, 2000; Lee, 2004; Popović and Ney, 2004; El-Kahlout and Oflazer, 2006; Oflazer and El-Kahlout, 2007; Bisazza and Federico, 2009; Salameh et al., 2015). Goldwater and McClosky (2005) insert pseudo-words derived from Czech source side morphology in places where the authors determine

that an English function word is needed.

Instead of segmenting the morphologically richer side, Ma et al. (2007) and Yeniterzi and Oflazer (2010) increase symmetry by joining consecutive words in the morphologically less complex side. When translating from a MRL, symmetry can be increased through morphological ***simplification***. Lee (2004) start from a fine-grained linguistic segmentation of the more complex side, automatically identifying morphemes to join or delete.

To the best of my knowledge, Sereewattana (2003) present the first use of ***unsupervised morphological segmentation*** in machine translation. The most popular subword segmentation method for MT at the time of writing appears to be BPE (Sennrich et al., 2015). Some other subword segmentation methods that have been applied to MT include Morfessor Baseline (Macháček et al., 2018), Morfessor CatMAP (Virpioja et al., 2007; de Gispert et al., 2009; Fishel and Kirik, 2010; Clifton and Sarkar, 2011), Morfessor FlatCat[15] (Ataman et al., 2017; Ataman and Federico, 2018; Banerjee and Bhattacharyya, 2018), SentencePiece (Kudo and Richardson, 2018; Kudo, 2018), and WordPiece (Schuster and Nakajima, 2012; Wu et al., 2016). In a related approach, Chitnis and DeNero (2015) use compression based on Huffman codes to represent rare words as sequences of common words.

When the target language 3 is represented using subword units, a desegmentation process is required for generating the final surface forms. In the most straightforward approaches, either word boundaries (*"a _wo rd"*) or subword boundaries (*"a wo +rd"*) are marked with a reserved symbol to enable reversing the segmentation. The boundary markers can attach to the previous or following token, or be separate tokens. Stymne and Cancedda (2011) and Cap et al. (2014) explore different strategies for a richer marking of word-internal token boundaries. The aim is to allow the translation system to produce compounds unseen in the training data, while imposing restrictions to discourage the generation of spurious or invalid compounds. Salameh et al. (2014) desegment by decoding from a lattice of morphs. Dyer et al. (2008) use a lattice to represent alternative segmentations of the input, allowing segmentation ambiguity to be retained. In neural architectures, lattice decoding is implemented in the Lattice LSTM (Sperber et al., 2017) and the Lattice Transformer (Zhang et al., 2019).

Another way of addressing segmentation ambiguity is to combine word-based and segmented systems using system combination (de Gispert et al., 2009; Virpioja et al., 2010; Srinivasan et al., 2019). Multiple different segmentations of the same data can be used. Pirinen et al. (2016) see a benefit from a system combination including Morfessor FlatCat. A third method that embraces segmentation ambiguity is subword regularization, discussed in Section 5.5.11.

As seen in Figure 5.2, the source 1 and target 2 3 do not need to use the same type of units. Methods making use of this freedom to increase symmetry by only segmenting one side were already discussed. It is also possible to intentionally

[15]Morfessor FlatCat was developed in the framework of this thesis.

use different granularities for input and output, e.g. the ***cross-scale*** system of Chung et al. (2016) uses BPE input and character output, while Costa-jussà and Fonollosa (2016) does the reverse with character input and word output. Decoder input 2 and output 3 are nearly always the same, although some exceptions exist (e.g. He et al., 2020). ***Multi-scale*** processing instead combines components operating on units of different granularity into a single system. It is possible to replace the source language embedding matrix with an embedding function computed by a neural network with the characters of the source word as input (Ling et al., 2015). Costa-jussà and Fonollosa (2016) apply this to the NMT encoder. Vylomova et al. (2017) experiment with other subwords as input, including Morfessor CatMAP. Morishita et al. (2018) compose embeddings from multiple segmentations with different-sized BPE vocabularies. The hybrid word–character decoder presented by Luong and Manning (2016) extends the multi-scale processing to the decoder side. First a word-level decoding is performed, with ⟨UNK⟩ generated for rare words. A second character-level decoder generates words to replace the ⟨UNK⟩s, conditioned on the state of the word-level decoder when generating that ⟨UNK⟩. The Helsinki NMT system (Östling et al., 2017), extended in Publication X, implements multi-scale processing on both encoder and decoder sides. Chung et al. (2016) base their decoder on a bi-scale recurrent unit, with a faster and a slower layer. The slower layer is updated only once the faster layer is done processing a particular subsequence, and is about to reset itself using a gate. On the encoder side, Cherry et al. (2018) apply the hierarchical multi-scale architecture of Chung et al. (2017) to learn a compressed sequence representation. Concurrently, Kreutzer and Sokolov (2018) apply the adaptive computation time architecture of Graves (2016) for the same purpose.

Concerning research into the role of morphology in machine translation, Bisazza and Tump (2018) perform a fine-grained analysis of how various source-side morphological features are encoded at different layers of the NMT encoder. They find that morphological information is only encoded in the later layers of the encoder, once context has been incorporated. Morphological information is not stored in the embeddings. This could be seen as an argument for treating morphology as a sentence-level phenomenon. Bisazza and Tump (2018) also find that morphological information is only captured to the extent that it is needed for accurately generating the target. This supports pretraining multilingual models with either related languages or a diverse set of languages. Belinkov et al. (2017) find that character-based representations are much better than word-based representations for learning morphology. The effect is especially prominent for low-frequency words. Voita et al. (2019) analyze multi-head attention, finding that only a small subset of attention heads appear to be important for the translation task, while the rest can be pruned without seriously affecting the performance. Important heads are specialized, having one or more interpretable functions, including attending to adjacent words and tracking specific syntactic relations.

Optimal granularity for machine translation

It has been shown that a linguistically correct segmentation does not coincide with the optimal segmentation for purposes of alignment and translation. In word alignment for SMT, Koehn and Knight (2003) find that the most linguistically accurate splits of German compounds do not result in the best SMT quality. Habash and Sadat (2006) improve on the linguistic segmentation using rule-based simplification, and Chung and Gildea (2009) through the use of statistical methods. Despite this, using unsupervised statistical segmentation methods for SMT has yielded mixed results. In many of these studies, unsupervised segmentation has worsened automatic scores compared to word-based translation (Virpioja et al., 2007; Fishel and Kirik, 2010; Rubino et al., 2015; Pirinen et al., 2016; Virpioja et al., 2010). The main benefit of segmentation has been a decrease in the ratio of untranslated words. In SMT, oversegmentation breaks words into units that are too small to carry meaning, and requires difficult many-to-many alignments. Word-level models with back-off to subword segmentation for OOV or rare words have been proposed (Yang and Kirchhoff, 2006; Pinnis et al., 2017). Wu et al. (2016) experiment with a back-off segmenting OOVs into characters in their mixed word–character model. Sennrich et al. (2017) resplit any rare subwords until the entire vocabulary exceeds a frequency threshold.

In NMT, the correspondence of the subwords to linguistic morphemes is even less important, as the encoders are able to determine the meaning of the units in context. Recent research (Cherry et al., 2018; Kreutzer and Sokolov, 2018; Arivazhagan et al., 2019b) indicates that smaller subwords are particularly useful for cross-lingual transfer to low-resource languages in supervised settings. Fully character-level translation has been approached using convolutional (Lee et al., 2017), recurrent (Cherry et al., 2018), and Transformer (Gupta et al., 2019) architectures. Even byte segmentation has been proposed (Costa-jussà et al., 2017). Chung et al. (2016) find that character-level decoders outperform subword-level decoders.

The downside of character-level models is that the sequences become very long, which can substantially impact training times. Also, unless downsampling is applied to reduce the sequence length in deeper layers, the depth of character-level models may be limited by memory use. Although including character-level models in the experiments of this thesis would have introduced useful points of comparison, due to the prohibitive cost of training, I chose to focus on subword models.

Unsupervised subword segmentation methods, tuned using criteria other than linguistic fidelity, such as the size of subword lexicon or the frequency distribution of the units, have become the standard in NMT. While vocabulary size tuning is typically performed using grid search, some automated approaches have been proposed. Salesky et al. (2020) automatically tune the BPE lexicon size for the task of NMT, by incrementally expanding the vocabulary with new subwords. Libovický and Fraser (2020) do the opposite by starting with a BPE subword lexicon and incrementally pruning until reaching a character model.

Table 5.1. Example from NMT system overfitted to the language modeling task.

Estonian	Source	Laktoosi puhul see nii ju ongi!
English	Overfit translation	I've been thinking about it.
English	Reference	That's the case with lactose!

5.3.2 Low-resource machine translation

Only a small subset of the world's languages have corpora of sufficient size to train state-of-the-art high-resource NLP systems. As parallel data is even more scarce than monolingual data, the resource problem is aggravated for the task of machine translation. This creates a need for machine translation methods designed specifically for the low-resource setting.

Data-driven MT systems are conditional language models. The training signal for the language model is much stronger than for the conditioning on the source. In a modern NMT system, the conditioning must pass through a natural language understanding encoder and a cross-lingually aligning attention mechanism. When a vanilla NMT system is trained in a low-resource setting, the learning signal may be sufficient to train the language model, but insufficient for the conditioning (Östling and Tiedemann, 2017b). In this case, the MT system degenerates into a fancy language model, with the output resembling generated nonsense, with possibly high fluency but little relation to the source text. As an example, Table 5.1 shows an output from an Estonian–English translation system trained from parallel data of only 18000 sentence pairs. Mueller et al. (2020) observe this language model overfitting phenomenon in a massively multilingual but low-resource setting using Bible translations as the corpus. High fluency is a known property of NMT (Toral and Sánchez-Cartagena, 2017; Koponen et al., 2019). The user is more likely to trust a translation if it looks like a proper sentence, and the user is unable to detect mistranslation without source language competence. Due to the risk of undetected misunderstanding, high fluency but low adequacy translations may score well under some evaluation settings, but still be highly undesirable in practice.

The challenges of low-resource settings exacerbate the challenges of morphologically rich languages, due to a combination of small data and a large vocabulary. The resulting data sparsity makes it difficult to estimate statistics for all but the most frequent items. Even though continuous-space representations allow neural methods to generalize well, they learn poorly from low-count events. Constructing the vocabulary using ***subword segmentation***[16] can reshape the frequency distribution of the basic units[17] to reduce sparsity, and yield a more balanced class distribution in the generator. Suitable subwords are also beneficial for ex-

[16] See Sections on vocabulary construction in MT (5.3.1) and subword segmentation methods (4.3).
[17] Illustrated in Figure 5.6 on page 137.

ploiting transfer from related high-resource languages (See Publication VIII) and monolingual data. Besides subword segmentation, the most commonly used approach to low-resource MT involves ***exploiting auxiliary data***, either parallel data for other language pairs (Section 5.3.3) or monolingual data (Section 5.3.4).

Other approaches include active learning and crowdsourcing (Ambati, 2012). Östling and Tiedemann (2017b) use external word alignments as supervision for learning sentence reordering. They translate one source token at a time, inserting the generated (possibly empty) target token anywhere in the output sequence using a position predictor. The NMT architecture and hyper-parameters can be optimized for the low-resource setting, e.g. by using smaller and fewer layers (Nguyen and Chiang, 2018) or by reducing batch size and increasing dropout (Sennrich and Zhang, 2019). An effective architectural change adds a lexical translation model to condition the generated output token directly on the aligned source embedding, without incorporating context (Nguyen and Chiang, 2018; Sennrich and Zhang, 2019). Nguyen and Chiang (2018) normalize target-side embeddings to remove frequency bias. Mueller and Lal (2019) adapt the model for each test sentence using a subset of similar training sentences.

Evaluations comparing NMT and PB-SMT under low-resource conditions have had mixed results, with some finding in favor of NMT (Bentivogli et al., 2016; Guzmán et al., 2019) and others (at least under some conditions) in favor of PB-SMT (Toral and Sánchez-Cartagena, 2017; Lample et al., 2018b; Artetxe et al., 2018a; Do Campo Bayón and Sánchez-Gijón, 2019). Popović (2017) finds that the strengths of PB-SMT and NMT are complementary, suggesting opportunities for hybridization. Koehn and Knowles (2017) identify worse quality in low-resource settings and out-of-domain as challenges for NMT systems. They also point out that even subword NMT shows weakness in translating low-frequency words belonging to highly-inflected categories, e.g verbs.

Unsupervised machine translation

In the most extreme low-resource setting, unsupervised machine translation, no parallel data at all can be used, only monolingual data from source and target languages. By considering parallel data to be labeled and monolingual data to be unlabeled, ***supervised MT*** corresponds to the low-resource training without auxiliary data described in Section 5.3.2, ***unsupervised MT*** uses no parallel data at all, and ***semi-supervised MT*** exploits a combination of small parallel data and large monolingual data.

A scenario without any parallel data, but still abundant monolingual data, is unrealistic in practice. Some small amounts of cross-lingual signal exists even for fairly low-resourced language pairs, e.g. Bible translations, a bilingual dictionary or even translated word lists and example sentences collected as language documentation. Languages lacking any cross-lingual resources are unlikely to have monolingual corpora of sufficient size and quality for the proposed unsupervised methods. Artetxe et al. (2020b) argue that the strict unsupervised scenario cannot be motivated from an practical perspective, but find other moti-

vations for research into unsupervised cross-lingual learning (UCL). There is an inherent scientific interest, in shedding light on theories concerning regularities of structure between languages, e.g. the distributional hypothesis. UCL provides a simplified, standardized lab setting to improve our understanding e.g. of embedding methods. Another use case deals with unknown or even non-human languages.[18] Finally, if almost the same quality as in semi-supervised methods is reachable, simple unsupervised methods may be preferable. An illuminating contrast can be made to the task of subword segmentation. The performance of unsupervised subword segmentation is often sufficient for applications, while unsupervised translation still lags behind semi-supervised MT. The optimal segmentation is not easy to quantify, and may vary according to the application, with unsupervised segmentation having desirable properties compared to the linguistically accurate segmentation. Unlike in translation, the supervision signal for segmentation is not a byproduct of human-to-human communication needs.

Early methods for unsupervised translation were based on applying decipherment to construct translation tables, which produce fluent target language output when applied to the source (Ravi and Knight, 2011; Dou and Knight, 2012, 2013). The breakthrough was transitioning to ***multilingual word embeddings***, trained on monolingual corpora and mapped into a shared embedding space. This mapping can be learned either based on a small seed dictionary of parallel words, or in an entirely unsupervised manner based only on the distributions. As a result of mapping embeddings into a joint space, a crude word-for-word translation system can be built. Using methods such as denoising autoencoding, back-translation, and adversarial learning, a better translation system can be bootstrapped from this crude initial system (Artetxe et al., 2018b; Lample et al., 2018a). More recently, pretraining of the entire network on denoising or masked language modeling tasks have yielded strong results (Conneau and Lample, 2019a; Song et al., 2019; Liu et al., 2020).

Unsupervised methods were first applied to NMT, but were also found to be applicable to PB-SMT (Lample et al., 2018b; Artetxe et al., 2018a). Also hybrid approaches combining NMT and SMT have been proposed (Marie and Fujita, 2018; Ren et al., 2019; Artetxe et al., 2019). Some of the unsupervised MT methods are also applicable to exploiting monolingual data in the semi-supervised MT setting (see Section 5.3.4).

In unsupervised NMT, cross-lingual transfer requires basic units to be aligned between languages without use of parallel data. When starting with pretrained embeddings, longer units are typically used, as they carry more meaning than short units, and are easier to align. In supervised NMT, a transition from early word-based methods to current subword-based methods has occurred. In unsupervised NMT, word-level methods are still in use.

[18] Although the rapid degradation of quality with increasingly distant language pairs raises concerns (Kim et al., 2020).

Zero-shot machine translation refers to an evaluation setting in which a multilingual MT system is evaluated on a language pair for which no parallel data was used at training time (Johnson et al., 2017; Ha et al., 2017; Lu et al., 2018; Blackwood et al., 2018; Vázquez et al., 2019; Aharoni et al., 2019; Arivazhagan et al., 2019a). The source and target languages are present in the training data individually, but not paired together. Zero-shot machine translation tests the ability of the system to generalize to unseen translation pairs.

Zero-resource translation is a related concept, which starts from a multilingual NMT system and extends it with an unseen translation language pair using synthetic pseudo-parallel corpora (Firat et al., 2016b; Currey and Heafield, 2019).

Asymmetric-resource machine translation describes a semi-supervised, multilingual MT learning setup in which the language pair of interest has much less resources than the auxiliary language pairs. The goal is to maximally exploit available resources, both parallel and monolingual, to improve the low-resource translation. Publication IX addresses asymmetric-resource machine translation into morphologically rich languages.

One of the main challenges in asymmetric-resource MT is the ***data imbalance***. In multi-task settings, the data imbalance is typically addressed by oversampling the low-resource data. One way to choose the oversampling weights is using a temperature-based approach to interpolate between sampling from the true distribution and sampling uniformly (Arivazhagan et al., 2019b). An alternative to oversampling the data is to adapt the gradient scale or learning rate individually for each task (Chen et al., 2018; Kendall et al., 2018). Scheduled multi-task learning[19] enables varying the oversampling rate during training.

5.3.3 Multilingual translation

Multilingual translation is the typical way of using cross-lingual transfer in MT. Multilingual translation can be viewed as a multi-task problem, in which each individual language pair forms a separate task. Seq2seq neural network architectures can be trained in multi-task settings, using the various parameter sharing techniques discussed in Section 3.4.2. Such a multilingual neural machine translation (MNMT) system has a difficult task, needing to learn a complex multi-way input-output mapping. Despite the difficulty of the task, multilingual training often results in improved performance. The improvement is due to an inductive bias: the learning signal from one language should help in learning other languages. Like other forms of transfer learning, multilingual translation has a regularizing effect. For general survey on multilingual translation, see Dabre et al. (2020). Arivazhagan et al. (2019b) also thoroughly review many of the challenges of multilingual translation.

Multilingual machine translation settings can be divided into three categories

[19] Presented in Section 3.4.2

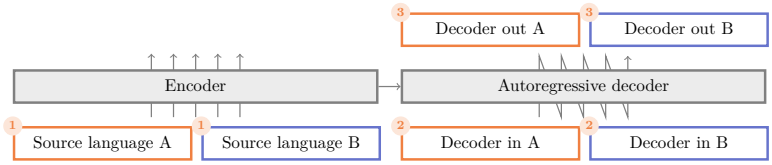


Figure 5.3. Recalling Figure 5.2, locations in an NMT system where different vocabularies can be used. The possibility of applying different subword segmentation models for each language is shown.

by cardinality: many-to-one, one-to-many, and many-to-many (Luong, 2016). ***Many-to-one*** translation involves many source languages but only one target language. The many-to-one setting can be described as a multi-domain learning problem. The target language is always the same, and the source language can be inferred from the input. It is thus not necessary to explicitly mark the languages involved in many-to-one translation. ***One-to-many*** translation is a classical multi-task problem. The system learns to perform two or more tasks on the same input. As the same input can be translated into any of the target languages, a mechanism for selecting the desired output is needed. In both many-to-one and one-to-many translation, English is typically selected as the language on the side with only one language, due to resource availability. ***Many-to-many*** translation combines the aspects of the two other settings. It is not necessary to have training data for all pairs of source and target supported by the many-to-many system. Missing language pairs can be evaluated in a zero-shot setting. Universal translation is the extension of many-to-many translation to cover all languages. The cardinality of the multilingual translation has an effect on the difficulty of the task: cross-lingual transfer is easier in the many-to-one setting compared to one-to-many (Arivazhagan et al., 2019b; Dabre et al., 2020).

There are several strategies for vocabulary construction in the multilingual setting. As seen in Figure 5.3, different vocabularies can be used for source 1, decoder input 2, and decoder output 3. Orthogonally to this choice, separate subword segmentation models can be applied to different languages in each of these locations. Cognate Morfessor in Publication VIII is an example of a system using separate decoder segmentations 2 3 for the two target languages. The most common choice is to use a ***joint vocabulary*** (Sennrich et al., 2015), a single multilingual representation for all locations and languages.

Some work emphasizes the role of subwords shared between different languages as anchors for a shared semantic space (Pires et al., 2019). However, there is recent work indicating a low importance of sharing subword vocabulary across languages (Artetxe et al., 2020a; Wu et al., 2019). These results indicate that cross-lingual generalization occurs on a level of linguistic abstractions, even without anchor subwords. Such linguistic abstractions may not be so readily found, e.g. word embedding spaces appear to be less isomorphic than thought, especially for distant pairs (Søgaard et al., 2018; Patra et al., 2019). Joint training can produce a more isomorphic word embedding space (Ormazabal et al., 2019).

In asymmetric-resource multilingual settings, training on a ***balanced data dis-***

tribution is important both for training of the translation system, and also for training the segmentation model. If the skewed distribution of available data is used as is, there is a risk that the units needed for the high-resource languages will be overrepresented in the vocabulary. Artetxe et al. (2020a) find that effective vocabulary size per language is an important predictor of performance. Balance can be achieved by resampling the data, or in the case of subword segmentation, by scaling the word counts of different languages.

Cross-lingual transfer is particularly useful between related languages, which share semantic, syntactic, and morphological regularities. Zoph et al. (2016) and Dabre et al. (2017) show that related parent languages result in better transfer. Liu et al. (2020) find that language transfer is more effective within similar language groups, but note that significant vocabulary sharing is not required for effective transfer. Contrary to the above, Kocmi and Bojar (2018) find in the case of Estonian that a larger parent (Czech) gave better results than a more related parent (Finnish). Arivazhagan et al. (2019b) report several findings concerning scaling to massively multilingual settings. As the number of tasks grows, performance degrades for all language pairs, especially the high and medium resource ones. However, the zero-shot performance increases when exceeding 100 languages. In massively multilingual settings, model capacity must be scaled up to remain sufficient. When increasing model capacity, adding depth is better than adding width.

Scenarios for using multilingual machine translation

Multilingual machine translation can be useful for reaching various goals. The most suitable methods depend on the scenario.

Low-resource multilingual NMT (LR-MNMT). Much of the literature focuses on leveraging high-resource language pairs to improve the performance on low-resource language pairs. Multilingual training improves low-resource and zero-resource language pairs in particular.

While multilingual translation to improve a low-resource translation direction has gained popularity in the NMT era, the technique was already proposed for PB-SMT (Nakov and Ng, 2009). Levinboim (2017) uses the property of transitivity to construct phrase tables for a low-resource pair though triangulation. A shallow translation e.g. using character or byte level models might be more appropriate than multilingual translation in the case when the source and target languages are closely related (Tiedemann, 2009; Costa-jussà et al., 2018).

Dong et al. (2015) proposes multilingual NMT through partial parameter sharing with a shared encoder and language-specific decoders. Using a simulated low-resource condition, they argue that the shared encoder helps in translating the low-resource pairs. Most of the work on low-resource multilingual MT uses sequential transfer (Zoph et al., 2016; Nguyen and Chiang, 2017). Passban et al. (2017) improves on the sequential transfer with a vocabulary adaptation step based on a bilingual dictionary. Murthy et al. (2019) find that pre-ordering the high-resource auxiliary parent to match the word order of the low-resource

child language is beneficial. Neubig and Hu (2018) and Gheini and May (2019) rapidly adapt to a new low-resource language, with transfer from a massively multilingual parent, trained ahead-of-time. Lakew et al. (2020) experiment on five low-resource African languages, finding parallel transfer to outperform single-pair, sequential transfer, and back-translation systems. Gu et al. (2018b) apply model-agnostic meta-learning (MAML) to low-resource NMT.

There is also work focusing on synthetic data and exploiting of monolingual data (Toral et al., 2019; Dabre et al., 2019; Valeev et al., 2019, and see Section 5.3.4).

Universal machine translation. The number of language-pair specific systems needed for universal translation between a set of languages scales with the square of the number of languages. For a large set of languages this number of systems quickly becomes infeasible, making multilingual translation appealing.

Supporting a large number of languages is easier in many-to-one translation, as the encoder can learn to recognize the source language without any language identifier (Lee et al., 2017). Using language-specific encoders or decoders is not suitable for universal translation, as the number of parameters scales linearly with a large number of additional parameters per language (Dong et al., 2015; Firat et al., 2016a). Full parameter sharing with a target language token is more amenable to scaling to massively multilingual settings (Johnson et al., 2017; Aharoni et al., 2019). The capacity of the model is still a limiting factor. In a massively multilingual model, interference from the other languages can degrade the performance, especially for high-resource languages (Arivazhagan et al., 2019b). This degradation is caused by limited model capacity being shared between too many tasks. Bapna and Firat (2019) take a step back from full parameter sharing, by adding small task-specific adaptor layers to a pretrained network, which only slightly increases the number of parameters for each added language. Methods for meta-learning which parameters to share also have a strong potential to scale (Platanios et al., 2018).

Scaling issues also affect the vocabulary of universal machine translation models. Different script systems and orthographic conventions pose challenges for universal translation, which requires representing the scripts of all languages equally well. Despite some lexical overlap between languages, word level vocabularies grow too rapidly, making subword or character level models more suitable. Ha et al. (2016) first use disjoint vocabularies using a source language coding attached to subwords, but later switch to a better scaling source language input factor (Ha et al., 2017).

Interlingual meaning representation. The distinction between work focusing on universal translation and interlingual meaning representation is fuzzy, as both goals are approached using massively multilingual models. Universal translation focuses on scaling the number of translation directions, while work on interlingual meaning representation focuses on the properties of the representations. The latter can use a smaller number of language pairs, but will include distant languages

with diverse semantics, as this can aid in learning a more abstract interlingual intermediate representation.

One approach attempts to construct the interlingual representation on the level of words. McCann et al. (2017) use the word embeddings from a single-pair NMT system as pretrained embeddings for downstream NLP. Gu et al. (2018a) propose a universal lexical representation, intended to embed the words of all languages in a single common space. The method is English-centric, assuming that words of all languages can be mapped to English words. However, languages vary in the amount of information carried by each word. Assigning the same size of representation for both the English article *"a"* and the Finnish verb *"istahtaisinkohan"* (Should I perhaps sit down for a moment?) seems unreasonable. To account for varying granularity, a truly universal representation must represent concepts rather than (e.g. whitespace delimited) words.

Another line of work explores fixed-size sentence representations found using MT. The FoTran project (Tiedemann et al., 2018) studies representations of this type. Schwenk and Douze (2017) and Espana-Bonet et al. (2017) use the bottleneck of a recurrent NMT architecture without attention as an advantage. The bottleneck vector can be used as an interlingual fixed-size sentence representation. Larger fixed-size representations can be constructed by inserting a cross-lingually shared attention bridge component between encoder and decoder (Lin et al., 2017; Vázquez et al., 2019; Lu et al., 2018).

The encoder of an attentional NMT system can be used to produce a variable-length sentence representation. If the encoder is not told the target language, it must produce a representation that works well for all target languages. Parameter sharing does not yet guarantee that the model learns a shared interlingual representation. If the target language is known to the encoder, it can encode the same source sentence into different target-language specific subspaces of the intermediate representation (Platanios et al., 2018). Lample et al. (2018a) force interlingual latent representations using an adversarial regularization term. The model tries to fool a discriminator which tries to identify the source language. The intermediate representations can also be regularized so that sentences with similar meaning in different languages are encoded into similar representations (Escolano et al., 2019a,b).

The zero-shot translation methods mentioned previously in Section 5.3.2, can also be placed in the category of LR-MNMT. However, as zero-shot quality benefits from language-invariant representations (Arivazhagan et al., 2019a), they belong equally well in the quest for interlingual representations.

Siddhant et al. (2020) evaluate the performance of sentence representations from massively multilingual NMT when used as pretrained features for downstream tasks. Kudugunta et al. (2019) make use of singular value canonical correlation analysis (SVCCA) (Raghu et al., 2017) to analyze representations from a massively multilingual NMT system. They find that representations of different languages cluster based on linguistic similarity. They also note a language-pair dependency in the representations: the encoder representations are dependent on

the target language and decoder representations on the source language. These results indicate that an interlingual meaning representation has not yet been reached.

5.3.4 Exploiting monolingual data

While parallel data is the primary type of data used for training MT models, methods for effectively exploiting the more abundant monolingual data can greatly increase the number of available examples to learn from. Use of monolingual data can be viewed as ***semi-supervised learning***: both labeled (parallel) and unlabeled (monolingual) data are used. There are two main approaches to exploiting monolingual data in MT: transfer learning and dataset augmentation.

Augmenting the data set with synthetic data
The easiest way to improve generalization is to train on more data. As natural training data is limited, a practical way to acquire more is to generate additional synthetic data for augmentation. The main benefit of ***dataset augmentation*** is as regularization to prevent overfitting to non-robust properties of small data.

Synthetic data can be self-generated by the model being trained, or generated by a related model. In machine translation, the best known example of synthetic data is ***back-translation*** (BT) (Sennrich et al., 2016). The process of back-translation begins with the training of a preliminary MT model in the reverse direction, from target to source. The target language monolingual data is translated using this model, producing a synthetic, pseudo-parallel data set with the potentially noisy MT output on the source side. Because the quality of the translation system used for the BT affects the noisiness of the synthetic data, the procedure can be improved by iterating with alternating translation direction (Lample et al., 2018b). Edunov et al. (2018) propose adding noise to the BT output. The benefit of noisy BT is further analyzed by Graça et al. (2019), who recommend turning off label smoothing in the reverse model when combined with sampling decoding. As a related strategy, Karakanta et al. (2018) convert parallel data from a high-resource language pair into synthetic data for a related low-resource pair using transliteration. Zhang and Zong (2016) exploit monolingual data through self-learning by "forward-translating" the monolingual source data to create synthetic parallel data. Self-learning has the benefit of not needing to train a second model, but the use of noisy synthetic data on the target side can be problematic.

Simple ways to generate synthetic data without a model include using a single dummy token on the source side (Sennrich et al., 2016), and copying the target to source (Currey et al., 2017). The latter can be interpreted as a target-side auto-encoder task without noise. The largest factor in determining the effectiveness of using synthetic data is how much the synthetic data deviates from the true data distribution. To avoid confusing the encoder with synthetic data from a different distribution than the natural data, it may be beneficial to use a special tag to

identify the synthetic data (Caswell et al., 2019).

Back-translation is based on the idea that translation should be invertible: if ***s*** translates into ***t***, then ***t*** should translate back into ***s***. ***Dual learning*** (He et al., 2016a; Cheng et al., 2016; Tu et al., 2017) takes this idea further by jointly training the forward and backward translation models, in a form of continuous BT.

Transfer learning from monolingual data

When using synthetic data, the aim is to produce data from a distribution that is as close as possible to the distribution of the main task. Transfer learning instead exploits different but related auxiliary tasks. Both approaches can make use of noise: if noise is added to parallel data it is called data augmentation, while denoising monolingual data is one of the loss functions used for transfer learning. Belinkov and Bisk (2017) apply both natural and synthetic noises for NMT evaluation, finding that standard character-based NMT models are not robust to these types of noise. Noise-based training methods have the benefit of making the model more robust to noisy inputs at decoding time.

Back-translation is a slow method due to the additional training of the reverse translation model. Transfer is a computationally cheaper way to exploit monolingual data as an auxiliary task.

The transfer from monolingual data is often sequential, in the form of ***monolingual pretraining*** of some of the parameters of the final translation model. The loss for the monolingual training can be different from the one used during NMT training.

The extent of the parameter transfer can vary. The smallest set of parameters is transferred when pretraining word (or subword) ***embeddings*** for the encoder, decoder, or both. Source and target embeddings can be pretrained on monolingual data from the source and target languages, respectively (Di Gangi and Federico, 2017). Alternatively, joint cross-lingual embeddings can be trained on both (Artetxe et al., 2018b). As the embeddings are trained for e.g. a generic contextual prediction task, this is a form of transfer learning. The pretrained embeddings can either be ***frozen*** or ***fine-tuned***, by respectively omitting or including them as trainable parameters during the NMT training. Thompson et al. (2018) investigate the effects of freezing various subnetwork parameters—including embeddings—on domain adaptation. Deep neural networks learn a hierarchy of features ranging from simple generic ones in earlier layers to more complex ones in later layers. This allows freezing parameters that are deemed most likely to be useful for other tasks. In the one-to-many setting, where the source language is shared between the tasks, freezing the parameters for the earliest layers of the network may be useful. This is particularly true for the source embeddings, which are the very first parameters to be applied.

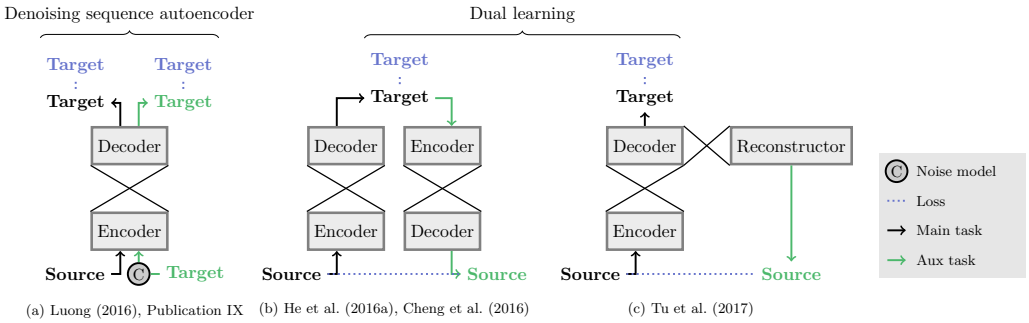
The parameters of a ***subnetwork*** of the translation model can be transferred: from a source language model to the encoder and from a target language model to the decoder. Ramachandran et al. (2017) pretrain the encoder and decoder with

source and target language modeling tasks, respectively. To prevent overfitting, they use task-mix fine-tuning: the translation and language modeling objectives are trained jointly (with equally weighted tasks). The work of Domhan and Hieber (2017) falls between transferring only embeddings and the whole decoder. They modify the NMT architecture by adding an auxiliary language model loss in the internal layers of the decoder, before attending to the source. This loss allows the first layers of the decoder to be trained on monolingual data. The ***full network*** can receive transfer from a sequence-to-sequence auxiliary task (Liu et al., 2020).

It is also possible to pretrain a separate language model for the target language, and combine its predictions with the ones of the translation model during decoding through ***language model fusion***. This approach is used in statistical machine translation, where one or more target language models are combined with a statistical translation model. It can also be applied in neural machine translation, through shallow fusion, deep fusion (Gulcehre et al., 2015), cold fusion (Sriram et al., 2017), or PostNorm (Stahlberg et al., 2018). Skorokhodov et al. (2018) use both pretraining (on both source and target side) and gated shallow fusion (on the target side) to transfer knowledge from pretrained language models. Some of the experiments are performed on low-resource data going down to 10k sentence pairs. An alternative way of using a language model is to perform a separate rescoring step on an n-best list or lattice. However, as a neural machine translation system is already a conditional language model, it may be preferable to avoid a separate language model and instead find a way to train the parameters of the NMT system using the monolingual data.

Loss functions for monolingual transfer are typically some form of language modeling loss. Some variants include the traditional next token prediction (Gulcehre et al., 2015), a masked language model (Song et al., 2019), a cross-lingual language model loss (Conneau and Lample, 2019b), or an autoencoder loss (Luong et al., 2015a). In spoken language translation, encoders are pretrained using automatic speech recognition auxiliary tasks, and decoders using textual MT (Bérard et al., 2018; Stoian et al., 2020).

While a unidirectional language model predicts the next token based on preceding context, a masked language model instead uses bidirectional context to fill in masked parts of the sentence, in a kind of cloze task. Devlin et al. (2019) mask out individual tokens. Joshi et al. (2020) and Song et al. (2019) mask out continuous spans of tokens, but replace the span with an equal number of mask tokens to retain the alignment between input and output. Lewis et al. (2019) replace arbitrary length spans of text (including zero length) with a single mask token, so that the model must predict how many tokens to produce in a particular context. As the input and output no longer have the same length, an encoder-decoder architecture is used. Song et al. (2019) mask the input tokens on the decoder side if they were unmasked in the encoder input, which forces the decoder to condition on the encoder instead of relying on decoder language modeling.



(a) Luong (2016), Publication IX (b) He et al. (2016a), Cheng et al. (2016) (c) Tu et al. (2017)

Figure 5.4. Alternative ways to set up an autoencoder auxiliary loss for NMT training.

The ***denoising sequence autoencoder*** (DSAE) frames language modeling as a denoising task. Target language text, corrupted by a noise model, is fed in as a pseudo-source. Different noise models can be used, e.g. applying reordering, deletions, or substitutions to the input tokens. The desired reconstruction output is the original noise-free target language text.

An autoencoder (Bourlard and Kamp, 1988) is a neural network that is trained to copy its input to its output. It applies an encoder mapping from input to a hidden representation, i.e. code $\boldsymbol{h} = f(\boldsymbol{x})$, and decoder mapping from code to a reconstruction of the input $\hat{\boldsymbol{x}} = g(\boldsymbol{h})$. To force the autoencoder to extract patterns in the data instead of finding the trivial identity function $\hat{\boldsymbol{x}} = \mathbf{1}(\mathbf{1}(\boldsymbol{x}))$, the capacity of the code must be restricted somehow. In the undercomplete autoencoder, the restriction is in the form of a bottleneck layer with small dimension. For example, in the original sequence autoencoder (Dai and Le, 2015), the entire sequence is compressed into a single vector.

In a modern sequence-to-sequence architecture, the attention mechanism ensures a very large bandwidth between encoder and decoder. When used as an autoencoder, the network is thus highly overcomplete. In this case, the capacity of the code has to be controlled by regularization. Robustness to noise is used as the regularizer in the ***denoising autoencoder*** (Vincent et al., 2008). Instead of feeding in the clean example $\boldsymbol{x}$, a corrupted copy of the input is sampled from a noise model $\mathbf{C}(\tilde{\boldsymbol{x}}|\boldsymbol{x})$. The denoising autoencoder must then learn to reverse the corruption to reconstruct the clean example. Masked language modeling can be seen as a special case of DSAE, where the noise model stochastically replaces some tokens with the mask symbol. Hill et al. (2016) use a DSAE[20] with an NMT-like architecture to learn fixed size sentence representations.

Comparing autoencoder setups for NMT. There are multiple ways of adding the autoencoder loss to the NMT training, three of which are shown in Figure 5.4.

The first setup (5.4a) treats the autoencoder task as if it was another language pair for multilingual training, and involves no changes to the architecture. When using this type of autoencoder task on target language sentences, the task cardinality changes into a many-to-one problem: the model must simultaneously learn

[20] Hill et al. (2016) use the term ***sequential denoising autoencoder*** (SDAE).

a mapping from source to target and from corrupted target to clean target. In both tasks the target language is the same. As the decoder is a conditional language model, this task strengthens the modeling of the target language. When using source language sentences, the model must simultaneously learn a one-to-many mapping from source to target and from corrupted source to clean source. Thus the decoder must learn to output both languages. The task may strengthen the encoder, by increasing its robustness to noise, and by preventing the encoding from becoming too specific to the target language. Luong et al. (2015a) and Luong (2016) experiment with various auxiliary tasks, including this type of autoencoder setup. They see a benefit of using the autoencoder task, as long as it has a low enough weight in the task mix.

Dual learning (5.4b) is a different way of adding an autoencoder loss. In dual learning the autoencoder is built from source-to-target and target-to-source translation models. He et al. (2016a) combine source-to-target and target-to-source translations in a closed loop which can be trained jointly, using two additional language modeling tasks (for source and target respectively), and reinforcement learning with policy gradient. Cheng et al. (2016) use a dual learning setup to exploit monolingual corpora in both source and target languages. Their loss consists of four parts: translation likelihoods in both directions, source autoencoder, and target autoencoder.

Tu et al. (2017) simplify the dual learning setup into an ***encoder–decoder–reconstructor*** network (5.4c). The reconstructor attends to the final hidden states of the decoder. As the reconstructor does not need a separate encoder, this removes the need for one subnetwork. One downside is that the training does not result in a target-to-source translation model. The aim of the work is to improve adequacy by penalizing undertranslation: the reconstructor is not able to generate any parts of the sentence omitted by the decoder.

5.4 Evaluation of machine translation

The translations in this thesis are evaluated using both human evaluation and several automatic metrics. The primary methods for human evaluation of translations are relative ranking, in which the evaluator places one or more translation hypotheses in order of preference, and direct assessment, in which each hypothesis is individually scored. Direct assessment can use a single quality scale, or decompose the assessment e.g. into fluency and adequacy. An alternative human evaluation is measuring the time required to post-edit the MT output. The WMT evaluation campaigns have used relative ranking and single scale direct assessment.

Human evaluation comes close to measuring the quality of translation in practice, with the remaining discrepancy caused e.g. by limited context during evaluation, and evaluator proficiency. Automatic metrics approximate translation quality imperfectly, but are nevertheless very useful due to much higher speed

and lower cost. Particularly during tuning of MT systems, frequent repeated evaluation is necessary.

Multiple automatic evaluation metrics are used to evaluate the work in this thesis. The most prominent is Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002). BLEU is computed as the geometric mean of precisions of word n-grams, with n ranging from 1 to 4, and each n-gram precision weighted by its own weight w_n.[21] To compensate for only considering precision but not recall, the score is modified by a brevity penalty to discourage too short hypotheses

$$\text{BLEU} = \overbrace{\min\left(1, \frac{\text{output-length}}{\text{reference-length}}\right)}^{\text{brevity penalty}} \left(\prod_{n=1}^{4} w_n \text{Precision}_n\right)^{\frac{1}{4}}. \tag{5.3}$$

BLEU assigns to each hypothesis a score between 0 and 1, typically expressed in percentage points, with higher scores being better. BLEU has been found to correlate well with human judgment for morphologically simple languages, when using a large enough number of references. Despite its inadequacies, BLEU is the standard evaluation metric in this field, and for that reason all the translation systems in this thesis are evaluated using it.

Other word-based metrics used in this thesis include METEOR and TER. METEOR (Banerjee and Lavie, 2005; Lavie and Agarwal, 2007; Denkowski and Lavie, 2011) computes an F-score from aligned stems of the reference and hypothesis, allowing for paraphrasing. To support these advanced features, METEOR requires knowledge-based resources for the target language. Translation Edit Rate (TER) (Snover et al., 2006) uses a shift operation that moves a contiguous sequence of words to another location, as well as a greedy search algorithm to find the minimum distance. TER does not require language resources, but the search algorithm is heavy.

A targeted evaluation, focusing on the ability of a model to translate rare and unseen words, is achieved using word unigram F_1-score (Sennrich et al., 2015), restricted to this class of words.

Rich morphology of the target language makes evaluation difficult. Word-level metrics score morphologically rich target languages lower overall. Analytical languages contain many non-inflecting high-frequency words (prepositions, articles), that give a larger number and longer n-gram matches. MRLs use complex word forms that are difficult to analyze and generate, while even a small change causes the entire word to be treated as an error. BLEU, TER, and many other word-based methods assume that a single word (or n-gram) is either correct or incorrect, nothing in between. This is problematic for inflected or derived words (e.g. "translate" and "translated" are considered two different words) as well as compound words (e.g. "salt-and-pepper" vs. "salt and pepper"). This is a minor issue for English, but it makes the evaluation unreliable for many other languages.

Character n-gram F-score (CHRF) (Popović, 2015, 2016) is a simple but well performing metric, which, due to operating on the character level, gives partial

[21] `mteval-v13a.pl` uses uniform weighting $w_n = 1 \; \forall \, n$.

	Method	Year	Pub.	$\|\boldsymbol{D}\|$	newstest 2015		newstest 2017	
					CHRF$_1$↑	BLEU ↑	CHRF$_1$↑	BLEU ↑
PB-SMT	word baseline	2015	VII	2.1M	—	10.0	—	—
PB-SMT	**tuned seg.**	2015	VII	2.1M	—	11.6	—	—
PB-SMT	word baseline	2016	VI	1.8M	—	10.48	—	—
PB-SMT	**ORM**	2016	VI	1.8M	—	11.21	—	—
NMT	**Morfessor FlatCat**	2017	X	1.8M	—	—	53.98	17.15
PB-SMT	Standard Moses baseline	2017	—	2.4M	—	—	55.93	15.9
NMT	**target-MTL**	2017	X	2.4M	—	—	57.57	20.31
NMT	monoling. Morfessor BL	2018	VIII	2.2M	57.94	20.87	61.33	23.11
NMT	BPE baseline	2018	VIII	2.2M (1M)	58.59	21.09	62.00	23.49
NMT	**Cognate Morfessor**	2018	VIII	2.2M (1M)	58.48	21.08	62.17	23.45

Table 5.2. Machine translation Character-F$_1$ (CHRF$_1$) and BLEU results for English→Finnish. Methods in boldface are contributions of this thesis. All methods are trained by the present author, except for the 2017 baseline, HY-SMT (A standard Moses phrase-based SMT system with BPE, back-translated news, CC-LM) by Östling et al. (2017). ↑indicates that higher scores are better. Parallel data sizes $|\boldsymbol{D}|$ do not include back-translation. For multilingual models, size of auxiliary ENG–EST data in parenthesis.

credit for hypothesis words that deviate from the correct form only by a few characters. It uses the F-score computed from the arithmetic mean of character n-grams with n ranging from 1 to 6.

5.5 Contributions to machine translation

In the following section, the contributions of this thesis with application to machine translation are discussed. First, the LEBLEU evaluation metric is presented, followed by results pertaining to use of subword segmentation or morphology, with general contributions to NMT at the end. Only machine translation evaluations are discussed here; for intrinsic evaluations of segmentation and details on the segmentation methods, see Section 4.4.

5.5.1 LeBLEU for evaluation of MT

Publication XII proposes LEBLEU[22] as an evaluation metric for morphologically rich target languages. LEBLEU extends BLEU score to consider fuzzy matches between word n-grams, based on letter edit distance from the Levenshtein (1966) algorithm.[23] A threshold is set for the maximal letter edit distance between hypothesis and reference n-grams, both to avoid giving credit for too dissimilar matches, and also to allow pruning to speed up the necessary edit distance com-

[22]Short for Levenshtein-BLEU or letter edit BLEU. Software available at `https://github.com/Waino/LeBLEU`.

[23]See Section 4.4.4 for the Levenshtein edit distance.

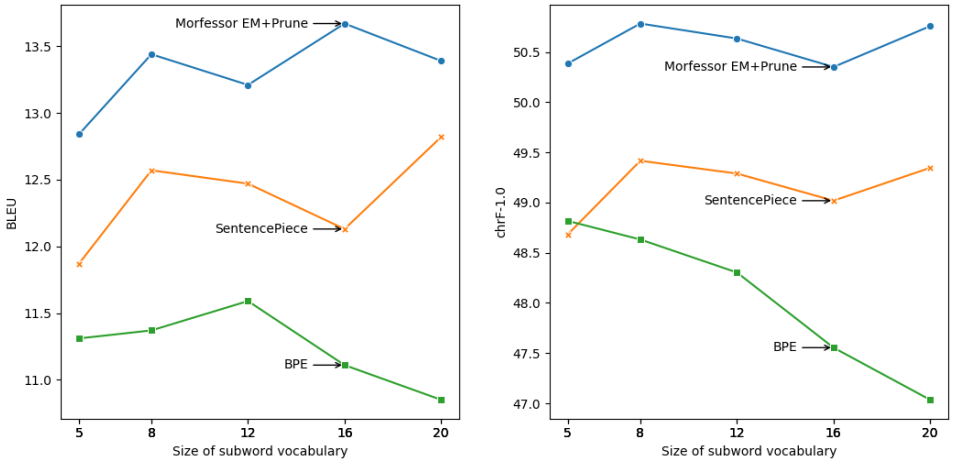


Figure 5.5. BLEU (left) and CHRF$_1$(right) results of varying the subword vocabulary in Publication IX. Multilingual models, with SRC+HRL+LRL autoencoder and full noise model, except for BPE which are multilingual models without autoencoder or noise. Results on English→Estonian newsdev2018.

putations between all pairs of *n*-grams. **(RQ2.1)** The results of Publication XII on WMT data sets show that fuzzy *n*-gram matching improves correlations to human evaluation, especially for highly compounding languages. LEBLEU has been used particularly for Indian languages (Kunchukuttan and Bhattacharyya, 2016; Murthy et al., 2019).

To avoid result tables growing too large, the number of evaluation metrics per paper had to be limited. CHRF is a simple, fast metric, and has gained some popularity. Looking only at translation directions *from* English, CHRF outperforms LEBLEU for both system- and segment-level evaluation for English→Finnish, English→Czech, English→Russian and the average of all WMT15 metrics task languages (Stanojević et al., 2015). In 2019, CHRF was still at the top, not significantly outperformed by any other metric for English→Finnish, English→Gujarati, English→Kazakh, and English→Latvian (Ma et al., 2019). In this thesis, CHRF was selected to complement BLEU as the primary tuning and evaluation metrics.

5.5.2 Overview of Morfessor methods in MT

Table 5.2 summarizes the automatic evaluation results for English→Finnish, which is the most frequently used translation direction in this thesis. The trend of improving performance over time is mainly due to general improvements in MT systems and available training data, and not only the contributions of this thesis. The quality increase from switching to the neural MT paradigm is notable. The reversion of performance in Publication VI is due to an error in LM training (for details, see the Publication). The problem affects all the evaluated systems and lowers the overall scores. However, it does not affect the increase in BLEU from the use of Omorfi-restricted Morfessor (ORM), verified using BLEU of a subset of the test set with all source sentences containing numbers removed.

Of the subword segmentation methods already presented in Section 4.4, some have been successfully applied to machine translation by other researchers. A lightly modified version of Morfessor FlatCat[24] was shown to outperform BPE for neural translation into English from Arabic, Czech, German, Italian, and Turkish (Ataman et al., 2017; Ataman and Federico, 2018); and Kazakh (Toral et al., 2019). For non-English target languages, Banerjee and Bhattacharyya (2018) showed improved results translating from English to Hindi and Bengali.

In Publication IX, Morfessor EM+Prune[25] was used to improve translation into low-resource morphologically complex languages. **(RQ1.2)** Figure 5.5 shows that Morfessor EM+Prune is superior to both SentencePiece and BPE in this asymmetrical-resource translation setting.

(RQ1.2) A small experiment was performed to analyze how subword segmentation affects the distribution of basic units. Figures 5.6 to 5.10 visualize the distributions of subwords from BPE, Morfessor EM+Prune and Morfessor FlatCat, comparing against the distributions of words and character n-grams. All five plots are computed from a 10M token subset of the Finnish Europarl corpus.

Figure 5.6 plots the frequency of basic units against the rank of the unit in a list sorted by frequency. The distributions of the subwords lie between those of words and short character n-grams, but the curves are more "boxy", i.e. the subword segmentations distribute the probability mass more uniformly over the entire lexicon. The long tail of rare items is truncated. To measure how much the distributions differ from the uniform distribution, Figure 5.7 shows the entropy of the distribution normalized by the logarithm of the size of the lexicon

$$-\sum_{i=1}^{|\mathrm{L}|} \frac{\big(P(m_i)\cdot \log P(m_i)\big)}{\log|\mathrm{L}|}. \tag{5.4}$$

Due to the normalization, the uniform distribution always receives the maximal score of 1, regardless of lexicon size. The closer the score is to 1, the closer the distribution is to the uniform distribution. A uniform distribution is desirable, because learners will typically overpredict the majority groups when trained on an imbalanced distribution (Krawczyk, 2016; Johnson and Khoshgoftaar, 2019). Steedman (2008) states that

> "[...] the machine learning techniques that we rely on are actually very bad at inducing systems for which the crucial information is in rare events [...]"

For the proportion of rare basic units occurring less than 10 times in the data set, shown by the dashed lines in Figure 5.8, optimal values are found for lexicons between 5k (BPE) and 20k (Morfessor EM+Prune). All subword segmentations are better at reducing rare units than the character n-grams. The Morfessor methods are better at alleviating sparsity than BPE.

[24]See Section 4.4.1 for Morfessor FlatCat.
[25]See Section 4.4.2 for Morfessor EM+Prune.

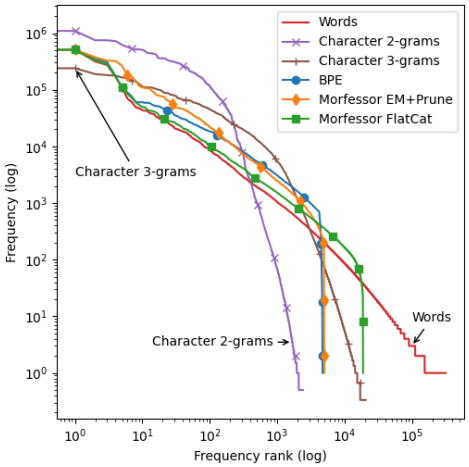


Figure 5.6. (On the left). Illustration of Zipf's law. Frequencies of units computed from a 10M token subset of the Finnish Europarl corpus. Character n-gram counts scaled down by n, to account for overlap. BPE and Morfessors are tuned for a lexicon size of 5k subwords.

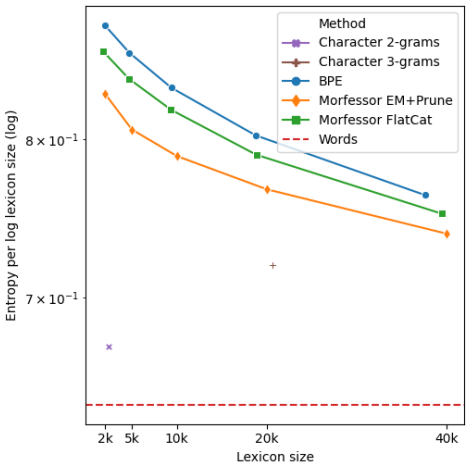


Figure 5.7. Divergence of subword frequencies from the uniform distribution, measured as entropy divided by log|L|. Larger values are closer to uniform.

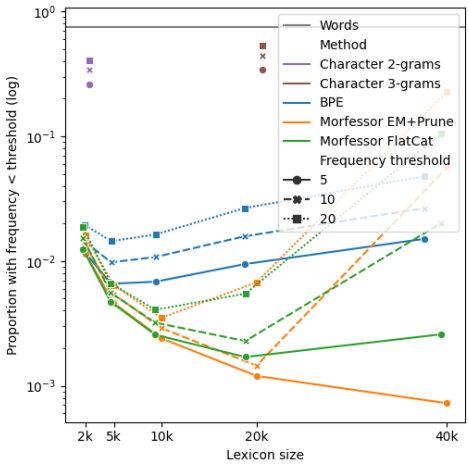


Figure 5.8. Proportion of the lexicon consisting of rare units, defined using three thresholds at 5, 10, and 20 occurrences.

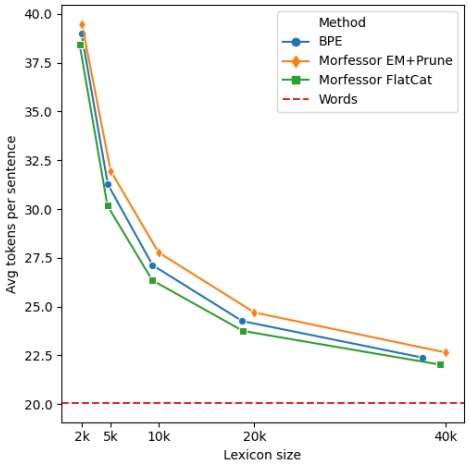


Figure 5.9. Average length of segmented sentences.

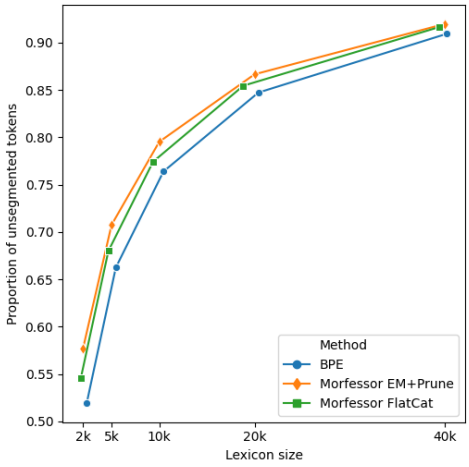


Figure 5.10. Proportion of unsegmented word tokens.

As a downside of small lexicons, Figure 5.9 shows that average sequence lengths grow rapidly when lexicons become small. For large lexicons, a substantial proportion of word tokens are not segmented at all, as shown in Figure 5.10. This means that subword models with large lexicons act essentially as word models with a fallback to subwords for some inputs.

To summarize, the distributions of BPE, Morfessor EM+Prune and Morfessor FlatCat subwords are somewhat similar when contrasted against words and character n-grams. All three methods produce segmentations that are closer to uniform than character n-grams. Smaller lexicon sizes bring the distributions closer to uniform, and decrease the proportion of rare items, but at the cost of producing very long sequences. BPE produces subword distributions that are closer to uniform, evidenced by slightly higher entropies. However, considering that the Morfessor methods outperform BPE in translation quality, it appears that other factors, such as the proportion of rare items or consistency of segmentation are more important than how close the distribution is to uniform.

5.5.3 Tuning the subword segmentation

Publication VII increases cross-lingual ***consistency between source and target*** representations via tuning the subword segmentation towards symmetric granularity. This is a form of very light ***cross-lingual transfer*** applied to segmentation. Only the morphologically rich Finnish target side is segmented, using Morfessor FlatCat.[26] The tuning aims for an equal number of Finnish subwords as there are English source words, by minimizing the sentence level absolute difference in token counts

$$\alpha^* = \arg\min_{\alpha} \sum_{(\boldsymbol{s},\boldsymbol{t})\in\boldsymbol{D}} \Big| |\boldsymbol{s}| - |\boldsymbol{M}(\boldsymbol{t};\alpha)| \Big|, \tag{5.5}$$

where $|\cdot|$ gives the number of tokens in the sentence, and $\boldsymbol{M}(\boldsymbol{t};\alpha)$ is the segmentation with a particular corpus likelihood weight hyper-parameter α. Balancing the token counts is a rough heuristic, and there is no guarantee that it increases the number of one-to-one alignments. **(RQ1.4)** Still, the tuned segmentation of Publication VII improves +1.1 BLEU over the word baseline in SMT.

Using a linguistically accurate morphological segmentation in PB-SMT is not an optimal choice. In general, oversegmentation seems to be a larger problem for statistical NLP applications than undersegmentation (Virpioja et al., 2011b). In the case of SMT, linguistic morphs may provide too high granularity compared to the second language, and deteriorate alignment (Habash and Sadat, 2006; Chung and Gildea, 2009; Clifton and Sarkar, 2011). Moreover, longer sequences of units are needed in the language model and the translation phrases to cover the same span of text. In NMT, oversegmentation is not as detrimental, but low-count events are poorly modeled. Subword lexicon size has been considered an important parameter to tune (Sennrich and Zhang, 2019; Ding et al., 2019;

[26] See Section 4.4.1 for Morfessor FlatCat.

Salesky et al., 2020). Ding et al. (2019) tune BPE segmentation granularity for English↔{Arabic, Czech, German, and French} NMT with datasets of approximately 200k sentence pairs, finding that optimal vocabulary sizes for Transformer architectures are very small, between character level and 4k subwords. For LSTM architectures, no single optimal granularity is found. Salesky et al. (2020) find optimal BPE vocabulary sizes between 20k and 40k subwords for GRU-based English→{Czech, German, and Turkish} translation with training datasets ranging between 105k and 208k sentence pairs. For English→Chinese with 227k training pairs the optimal size is 10k subwords.

(RQ1.1) The optimal subword vocabularies were 64k in Publication VIII, 50k in Publication XI, and 16k in Publication IX. In higher-resource settings the performance is not as sensitive to vocabulary size, and the optimal subword vocabulary size is larger. A preference for larger vocabularies is caused by transfer at the level of basic unit embeddings, e.g. when optimizing segmentations for cross-lingual consistency. Very small units do not carry meaning, making longer units more suited for transfer at the embedding level. Use of shallow networks also leads to a preference for larger vocabularies, as less computation is available for encoding in context. In low-resource settings and related target languages, optimal vocabularies are smaller. Introducing subword regularization in Publication IX lessens the impact of the subword vocabulary size, as shown in Figure 5.5 on page 135. Preliminary experiments of low-resource NMT without subword regularization suggested a more substantial effect for the lexicon size.

5.5.4 Restricted segmentation

Oversegmentation is detrimental for SMT, due to the hard limits on the spans of context considered. As a consequence, SMT using unsupervised morphological segmentation with e.g. Morfessor CatMAP has typically lead to improvements in OOV rate, but not overall quality (e.g. Virpioja et al., 2007). Omorfi-restricted Morfessor[27] improves language-internal consistency by combining the strengths of rule-based and data-driven segmentation. **(RQ1.2)** As shown in Publication VI and Table 5.2, using Omorfi-restricted Morfessor in SMT does improve over both the word baseline and the Omorfi segmentation. As the method results in a large lexicon and is unable to address rare or OOV stems, it is not well suited for use in NMT. For relevance in the NMT era, potential future work could extend ORM to allow two types of deviation from the linguistic segmentation: (i) the already implemented forming of supermorphs from two or more linguistic morphs, and (ii) splitting a rare morph into submorphs, in such a way that none of the submorphs are allowed to cross a linguistic morph boundary. For example, given the linguistic segmentation *"München+i+stä+kin"* (also from Munich), *"Mün+ch+en+istäkin"* would be an allowed segmentation, but *"Münchenis+täk+in"* would not.

[27] See Section 4.4.3 for Omorfi-restricted Morfessor.

	Method	Year	Pub.	$\|\boldsymbol{D}\|$	newsdev 2018		newstest 2018	
					CHRF$_1$↑	BLEU ↑	CHRF$_1$↑	BLEU ↑
NMT	BPE baseline	2018	VIII	1M (2.2M)	56.52	17.93	—	—
NMT	**Cognate Morfessor**	2018	VIII	1M (2.2M)	57.05	18.40	—	20.7
NMT	Standard back-translation	2020	IX	18k (–)	—	—	36.12	5.51
NMT	**Scheduled MTL**	2020	IX	**18k** (19.4M)	—	—	56.45	18.05

Table 5.3. Machine translation Character-F$_1$ (CHRF$_1$) and BLEU results for English→Estonian. Methods in boldface are contributions of this thesis. The standard back-translation baseline uses the same segmentation method (Morfessor EM+Prune) as the method using scheduled MTL, but without subword regularization. The baseline does not use multilingual training or DSAE, but uses back-translation. ↑indicates that higher scores are better. Parallel data sizes $|\boldsymbol{D}|$ do not include back-translation. The size of the auxiliary ENG–FIN data is shown in parenthesis, after the size of the ENG–EST data.

5.5.5 Boundary correction

One benefit of segmented translation is the ability to generate new compounds and inflections that were not seen in the training data. This ability is important for translation into languages with productive morphology. However, the productive generation ability can also lead to errors, e.g when an English word frequently aligned to a Finnish compound modifier is translated using such a morph, even though there is no compound head to modify. The "dangling" morph boundary marker will then cause the space to be omitted, forming an incorrect compound with whatever word happens to follow. For example, the Finnish pronoun *"moni"* (many) is also a frequent prefix, as in *"monitoimi-"* (multi-purpose) or *"monikulttuurinen"* (multicultural). Confusing the two may result in an erroneous novel compound in *"*moniliberaalien keskuudessa"* (among the *multiliberals). To correct the error, a space should be introduced between *"moni"* and *"liberaalien"*, leading to a correct translation (many among the liberals). In the opposite type of error, compounds may be translated as separate words, or hyphenated compounds translated with the hyphen omitted.

Publication VI proposes a neural boundary predictor for correcting this type of error. While the predictor works reliably for the noisily boundary marked but otherwise correct Finnish text it was trained on, manual inspection shows that the performance is erratic for disfluent translation output.

The mispredicted boundary markers are no longer a problem for NMT systems, due to their strong target language modeling ability. As mentioned in Section 5.3.2, PB-SMT has been found to be competitive in some low-resource settings. Perhaps a strong, modern tagger such as LaserTagger (Malmi et al., 2019) could be effective in boundary correction for low-resource PB-SMT. LaserTagger is able to perform editing tasks, such as sentence fusion and splitting, by tagging each token with one of KEEP, DELETE or adding one of a small set of predetermined joining phrases before the token.

5.5.6 Cross-lingual transfer using Cognate Morfessor

The aim of Cognate Morfessor[28] is to increase target–target consistency in the asymmetric-resource multilingual translation task. Cross-lingually consistent segmentation finds subwords with similar surface forms and meanings across languages, improving cross-lingual transfer on the level of subword representations. If segmentation decisions are consistent between the high- and low-resource target languages, better representations for the units in the low-resource language can be learned using the contexts of their correspondents in the high-resource language.

(RQ1.4) The experiment in Publication VIII was multilingual, with two medium-resource language pairs: English→Finnish with 2.2M sentence pairs of natural data, and English→Estonian with 1M sentence pairs. Table 5.2 shows that for the higher-resourced ENG→FIN pair, multilingual training with Cognate Morfessor improves translation quality over monolingual Morfessor Baseline, but results are inconsistent when comparing against BPE-segmented multilingual training. Table 5.3 shows that for the lower-resourced ENG→EST the cross-lingual segmentation is clearly beneficial, yielding a +0.47 BLEU and +0.53 CHRF$_1$ increase compared to BPE. One downside is that, due to the model structure, Cognate Morfessor is currently not applicable to more than two target languages.

5.5.7 Target side multi-task learning

In a hybrid word–character decoder (Luong and Manning, 2016), the word level decoder outputs frequent words as they are, and outputs ⟨UNK⟩ for rare words, while storing the decoder hidden state at that timestep. A second character-level decoder expands the ⟨UNK⟩s into surface forms, conditioned on the stored state. As a downside, the morphological realization of the rare word is not yet decided during the word-level decoding, which can be problematic for long-range morphological dependencies, such as agreement.

Publication X implicitly incorporates morphological information into NMT through target-side multi-task learning (MTL). Labels produced by the FinnPos morphological analyzer (Silfverberg et al., 2016) provide additional supervision signals that can help the model learn target-side grammar and morphology. Morphological analyzers and parsers typically produce labels on the level of words. Target-side MTL requires the label sequence to be of the same length and synchronous with the surface sequence. This is problematic for subword decoders, as labels need to be repeated and/or distributed over the subwords. The hybrid word–character decoder makes it simple to use word-level labels, while still supporting open vocabulary translation and avoiding unwieldy large model vocabularies.

The degree of parameter sharing in target-side MTL is very high. In addition to sharing the encoder, all parts of the word-level decoder except the final prediction

[28] See Section 4.4.4 for Cognate Morfessor.

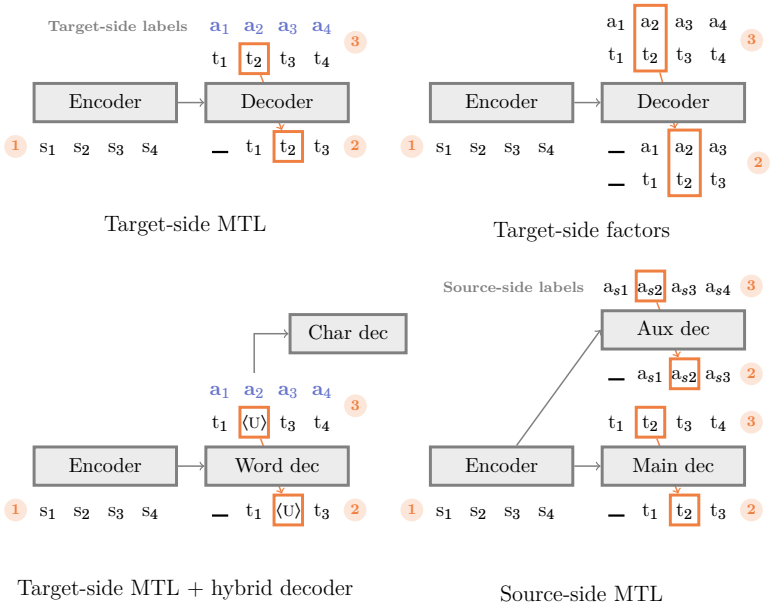


Figure 5.11. Ways of using auxiliary labels in NMT. s_i are source tokens, t_i target tokens, a_{si} source labels, and a_i target labels.

layers are shared. Although transfer is more commonly used to strengthen the encoder (Luong et al., 2015a; Niehues and Cho, 2017), there is other work exploring target-side auxiliary tasks (Dalvi et al., 2017; Eriguchi et al., 2017; Nadejde et al., 2017) concurrently with Publication X.

Figure 5.11 visualizes how target-side MTL resembles using a factored representation on the target side. The difference is that in a factored model, the labels are present in both decoder input 2 and decoder output 3, while in target-side MTL they are only present in the output 3. When training a system using factored output, embedded gold standard labels are given as input to the decoder. During translation gold standard labels are not available, and predicted labels are instead fed back in, without accounting for the confidence of the predictions. This might worsen the problems caused by exposure bias, i.e. the mismatch between the data distributions during training (samples from the data) and inference (samples from the model) (Ranzato et al., 2016). In terms of computational cost, a factored model needs to predict the auxiliary labels also during translation, slowing down inference and complicating the beam search. A factored model might also need to use a larger beam to avoid hypotheses with the same surface form but different labels from crowding out more diverse hypotheses. In target-side MTL, the auxiliary tasks are only performed during training, and no changes need to be made to the inference.

Based on an ablation experiment, target-side multi-task learning improves BLEU by +2.03. PoS-tags and lemma clusters are important labels, but the morphological tags are not. Not all auxiliary tasks are useful in MTL (Bingel and Søgaard, 2017; Bjerva, 2017). Belinkov et al. (2017) find that neural decoders learn very little about word structure, as the attention mechanism removes much of the

burden of learning word representations from the decoder. These results indicate that syntactic supervision for the decoder is more useful than morphological supervision.

A common type of error made by the system[29] described in Publication X is overtranslation through repetition. A possible explanation for the effect is the lack of connection from the character-level decoder back to the word-level. The architecture lacks this connection due to technical limitations of Theano. The surface forms generated by the character-level decoder are conditionally independent given the word-level hidden states, which can be similar to the states at adjacent timesteps. The lack of this connection makes the architecture less well suited to a morphologically rich target language, as the information loss becomes more severe with increasing proportion of ⟨UNK⟩s.

In subsequent related work, (Passban et al., 2018) use a target-side MTL approach by incorporating an auxiliary prediction channel in a character-based decoder. The auxiliary label is STM for characters belonging to the stem, or the identity of the affix for characters in affixes.

5.5.8 Overattending penalty

In neural machine translation, adequacy is reduced by two complementary issues: undertranslation and overtranslation. In undertranslation, a part of the source sentence is left untranslated, producing a shorter and less informative output. In overtranslation, the translation of a part of the source sentence is included several times in the output. A common error combines both, e.g. *"Tapasin Bobin ja Huldericin"* ↦ *"I met Bob and Bob"*, in which the common name *"Bob"* is overtranslated and the rare name *"Hulderic"* is undertranslated. Overtranslation also occurs when the decoder becomes stuck producing the same token over and over *"I met Bob and and and…"*.

To reduce undertranslation, Wu et al. (2016) modify the beam search scoring function[30] by adding two heuristic penalties: length normalization (lp) and coverage penalty (cp)

$$\mathrm{lp}(\boldsymbol{t}) = \frac{(|\boldsymbol{t}| + \lambda)^{\alpha}}{(1+\lambda)^{\alpha}}; \qquad \mathrm{cp}(\boldsymbol{t}, \boldsymbol{s}) = \beta \sum_{i=1}^{|\boldsymbol{s}|} \log\left(\min(\sum_{j=1}^{|\boldsymbol{t}|} \alpha_{ij}, 1.0)\right). \tag{5.6}$$

where the parameters α, β, and λ control the strengths of the penalties, and α_{ij} are the attention weights. Huang et al. (2017) propose an alternate bounded length reward. Beam search penalties change the way in which the decoding is performed, but do not change the model. Reducing undertranslation has also been addressed by extending the model itself with coverage tracking components inspired by fertility mechanisms in SMT (Tu et al., 2016), and by dual learning (Tu et al., 2017).

[29] Software available at `https://github.com/Waino/hnmt`.
[30] Beam search described in Section 3.5.

Publication X proposes an overattending penalty (oap)

$$\mathrm{oap}(\boldsymbol{t},\boldsymbol{s}) = -\gamma \max\left(\max_{i=1}^{|\boldsymbol{s}|}\left(\sum_{j=1}^{|\boldsymbol{t}|} a_{ij} - 1.0\right), 0.0\right). \tag{5.7}$$

The γ parameter controls the strength of oap. The penalty is applied if the most attended source word has sum attention over 1.0, i.e. the decoder has attended to it more than is necessary to output a single target token. The overattending penalty is monotonically increasing, which enables using it for early pruning of active hypotheses. The overattending penalty is specifically designed to alleviate an observed tendency of overtranslation in the hybrid word–character architecture. It also requires a translation direction with a granularity asymmetry, in which target sequences are generally shorter than source sequences. It is not appropriate in the case where the decoder must emit several target tokens per source token, e.g. in subword models.

A beam search scoring function incorporating all three penalties is given by

$$\mathrm{score}(\boldsymbol{t},\boldsymbol{s}) = -\frac{\log\left(p(\boldsymbol{t}\,|\,\boldsymbol{s})\right)}{\mathrm{lp}(\boldsymbol{t})} + \mathrm{cp}(\boldsymbol{t},\boldsymbol{s}) + \mathrm{oap}(\boldsymbol{t},\boldsymbol{s}). \tag{5.8}$$

5.5.9 Segmenting proper names in multi-scale NMT

One problem with the hybrid word–character encoder–decoder (Luong and Manning, 2016) is the handling of rare copyable words, e.g. proper names. The architecture is able to translate them, using the character-level encoder to represent, and the character-level decoder to generate. However, the information flows through two single-vector bottlenecks in the word-level encoder and decoder respectively, and the word-level attention mechanism is not able to alleviate the bottleneck.

Publication X addresses the problem by adding an attention mechanism to the character-level decoder, combined with character segmentation of names. To enable copying or transcription on a subword-level, the attended sequence needs to be modified to contain the subword-level information. A simple solution involves segmenting proper names into characters. Proper names are approximated using a rough heuristic: any token longer than one character beginning with an upper case letter or digit, after true-casing. All segmented characters are marked using reserved symbols, similar to morph boundary marking. The first and last characters of the sequence have distinct symbols separating them from word-internal characters, and enabling reversible segmentation. The segmentation procedure is inspired by Wu et al. (2016), who use a similar segmentation in their mixed word–character model, which does not use multi-scale processing.

The combination of adding the attention mechanism in the character decoder and segmenting names into characters yields an improvement of +4.3 BLEU for English→Finnish translation with the hybrid word–character encoder–decoder architecture. While the proposed method is not directly applicable outside of

			Autoencoder			ENG–EST			ENG–DAN			ENG–SLO		
Method	ML	BT	SRC	HRL	LRL	chrF1	BLEU	rare	chrF1	BLEU	rare	chrF1	BLEU	rare
ML, AE, full BT	✓	✓	✓		✓	56.45	18.05	41.13	51.27	14.80	56.63	52.80	16.87	70.97
ML, no AE, full BT	✓	✓				56.33	18.15	40.85	51.20	15.00	57.39	52.65	16.63	70.82
ML, AE, no BT	✓		✓		✓	51.71	14.04	34.79	50.06	13.92	54.58	50.19	14.02	69.94
Only ML	✓					50.09	12.90	33.20	49.57	13.13	54.21	49.83	13.97	68.79
Only AE			✓		✓	42.65	8.19	21.59	42.26	7.60	44.48	38.97	6.25	62.51
Vanilla BT		✓†				36.12	5.51	13.25	—	—	—	—	—	—
Vanilla						29.46	2.64	6.22	31.95	2.63	30.40	23.76	1.27	36.80

Table 5.4. Results for cross-lingual transfer (ML for multilingual), back-translation (BT), and denoising sequence autoencoder (AE). ✓† indicates the use of a low-quality back-translation made with a non-multilingual non-autoencoder vanilla BT model.

hybrid multi-scale models, the result does highlight the importance of copying for rare proper names, a challenging class of words.

5.5.10 Multilingual NMT

(RQ3.2) Multilingual training is very effective in the settings examined in this thesis. The benefit is most prominent for the language pairs with fewer resources. In Publication VIII, the lower-resourced ENG→EST pair benefits +2.6 BLEU. In Publication IX, the largest gains of up to +12.7 BLEU come from cross-lingual transfer. In Publication VIII, even the higher-resourced ENG→FIN pair benefits slightly from the cross-lingual transfer, with improvements up to +0.5 BLEU for 3 of 4 test sets, and -0.1 BLEU for the fourth test set. **(RQ3.3)** In accordance with Zoph et al. (2016) and Dabre et al. (2017) and in contradiction with Kocmi and Bojar (2018) the results of Publication IX support the conclusion that language relatedness is more important than parent dataset size. Monolingual fine-tuning[31] consistently improves results for both language pairs in Publication VIII. Fine-tuning is also beneficial in Publication IX, but due to the very small data, fine-tuning on a mix of tasks is important, and care must be taken to select task weights to avoid overfitting.

The data conditions in these experiments—with a small number of related languages and a very small amount of parallel data for the target language of interest—are well suited for increasing cross-lingual transfer and minimizing interference. The number of language pairs is not large enough to cause model capacity issues, and data for a single pair is not large enough to make cross-lingual transfer ineffective.

5.5.11 Asymmetric-resource transfer learning

Publication IX focuses on an asymmetric-resource multilingual MT scenario, in which the goal is to improve translation into a low-resource language (LRL),

[31] Using the mixed pretraining setup described in Section 3.4.2.

Method	CHRF$_1$	BLEU	rare
3-phase scheduled multi-task	51.71	13.94	33.96
2-phase scheduled multi-task	51.42	13.75	33.83
Multi-task w/o schedule	48.62	11.98	29.16
HRL pretraining, LRL fine-tuning	48.15	11.35	29.88

Table 5.5. Results for scheduled multi-task learning. English-Estonian. All models are multilingual multi-task models using auxiliary autoencoder tasks in three languages: source, high-resource target (HRL), and low-resource target (LRL) language. No back-translation is used.

by maximally exploiting the available resources. The resources include a small amount of parallel data for the SRC–LRL language pair of interest, and much larger auxiliary parallel data between the same source and a high-resource language (HRL), which is related to the LRL. Additionally there is monolingual data for all three languages. The monolingual data is exploited in two different ways: using a ***denoising sequence autoencoder*** (DSAE)[32] task, and using back-translation (BT).[33] As a consequence, the setting involves a large number of heterogeneous tasks, which can be grouped by the type of loss (translation or DSAE), the origin of the data (natural or synthetic), and the amount of resources (HRL or LRL).

(RQ3.2) Table 5.4 summarizes results on using cross-lingual transfer, back-translation, and autoencoder. The largest individual gains of up to +12.7 BLEU come from cross-lingual transfer. **(RQ3.4)** For exploiting monolingual data, the DSAE on its own results in up to +5.5 BLEU improvement. Back-translation is only effective when combined with the other techniques. When using only back-translation, i.e. back-translating the data with a weak model trained only on the low-resource parallel data, and then training a forward model augmented only by this low-quality back-translation but not multilingual training or autoencoder, the performance is very low: only +2.87 BLEU better than the vanilla model without back-translation. The high-quality back-translation together with multilingual training gives an +12.7 BLEU increase over the vanilla back-translation. Combining all the techniques together results in gains of up to +15.6 BLEU. **(RQ3.1)** For the vanilla model, reducing the size of the network and the minibatches resulted in improved performance. However, training a large model with the additional auxiliary data resulted in much better performance. As seen in Table 5.3 on page 140, this result reaches within a few BLEU points of previous medium-resource results, while using only 18 000 sentence pairs. **(RQ4.1)** Figure 5.12 shows how varying the amount of low-resource data affects the translation quality.

[32] See Section 5.3.4 for denoising sequence autoencoder.

[33] Software available at `https://github.com/Waino/OpenNMT-py/tree/dynamicdata`.

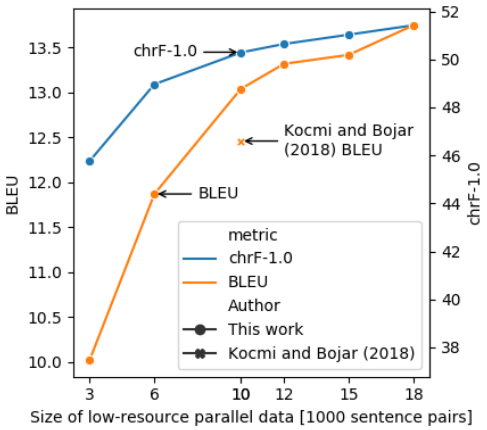


Figure 5.12. Varying the amount of low-resource data. Multilingual models, with SRC+HRL+LRL autoencoder and full noise model. Results on English→Estonian newstest2018.

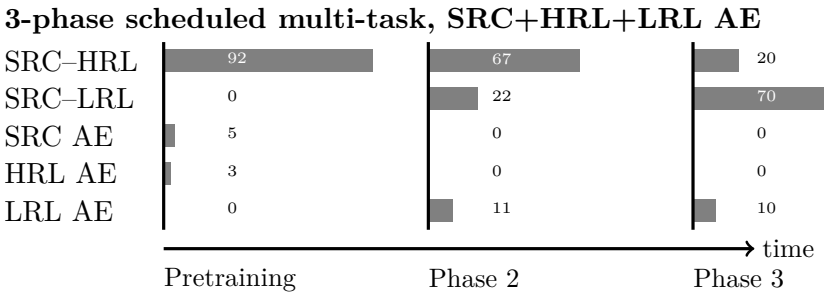


Figure 5.13. The task mix schedule used in the 3-phase scheduled multi-task learning experiment. The 2-phase schedule is the same, except it omits the third phase, continuing the second phase until the end of training.

Scheduled multi-task learning

Recall from Section 3.4.2 that fully sequential transfer may suffer from catastrophic forgetting, and fully parallel transfer requires careful tuning of task-mix weights when the amounts of data are imbalanced. ***Scheduled multi-task learning*** allows a compromise between the two extremes. The schedules proposed by Kiperwasser and Ballesteros (2018) are designed for the two-task case. A new partwise constant task-mix schedule suitable for an asymmetric-resource setting with multiple, heterogeneous auxiliary tasks is proposed in Publication IX. The task-mix schedule can have an arbitrary number of steps, any of which can be mixing multiple tasks. As an example of a task-mix schedule, the 3-phase schedule is shown in Figure 5.13.

Table 5.5 shows evaluations for different configurations of transfer learning. The system marked *HRL pretraining, LRL fine-tuning* uses a mix of HRL translation and autoencoder tasks for pretraining, and only a single task—LRL translation—for fine-tuning, and is thus fully sequential in terms of languages. It quickly overfits in the fine-tuning phase, resulting in the weakest performance. Fully parallel transfer is achieved by multi-task learning without schedule, trained with a constant task mixing distribution.

(RQ3.6) The models using scheduled multi-task learning effectively combine sequential and parallel transfer, resulting in improved performance. In 2-phase scheduled multi-task, LRL tasks are not used in the pretraining phase, but a mix of tasks is used for fine-tuning. It gives a benefit of +2.4 BLEU compared to the model fine-tuning on only LRL tasks, and +1.77 BLEU compared to training with a constant mixing distribution. The 3-phase scheduled multi-task adds a third phase training mostly on LRL tasks. A small proportion of HRL translation is included to delay overfitting. The model again overfits in the final phase, but does reach a higher score before doing so.

Noise model for text

To apply a denoising sequence autoencoder to text, a suitable noise model is needed. In domains such as image and speech, there are very intuitive noises, including rotating, scaling, and mirroring for images; and reverberation, time-scale stretching, and pitch shifting for speech. As text is a sequence of discrete symbols, where even a small change can have a drastic effect on meaning, suitable noise models are less intuitive. It is not feasible to guarantee the noise does not change the correct translation of the input.

Several noise models for text have been proposed in the literature. The contributions of this thesis add two new noises: word boundary noise and taboo sampling segmentation.

Local reordering. Lample et al. (2018a) perform a local reordering operation σ that they call ***slightly shuffling*** the sentence. The reordering is achieved by adding to the index i of each token a random offset drawn from the uniform distribution from 0 to a maximum distance k. The tokens are then sorted according to the offset indices. This maintains the condition $\forall i \in \{1, n\}, |\sigma(i) - i| \leq k$. Instead of shuffling tokens, Lewis et al. (2019) perform sentence permutation by randomly shuffling the order of the original sentences in a document-level model.

Token deletion. Randomly dropping tokens is perhaps the most commonly used noise. It is the central idea in ***word dropout*** (Iyyer et al., 2015), in which each token is dropped according to a Bernoulli distribution parameterized by a tunable dropout probability.

Token insertion. Randomly selected tokens can also be inserted into the sentence. The tokens can be sampled from the entire vocabulary, or from a particular class of tokens. E.g. Vaibhav et al. (2019) insert three classes of tokens: stop words, expletives, and emoticons.

Token substitution. SwitchOut (Wang et al., 2018) applies random substitutions to tokens both in the source and the target sentence. One benefit of SwitchOut is that it can easily and efficiently be applied late in the data processing pipeline, even to a numericalized and padded minibatch. Any noises that affect the length of the sequence are best applied before numericalization.

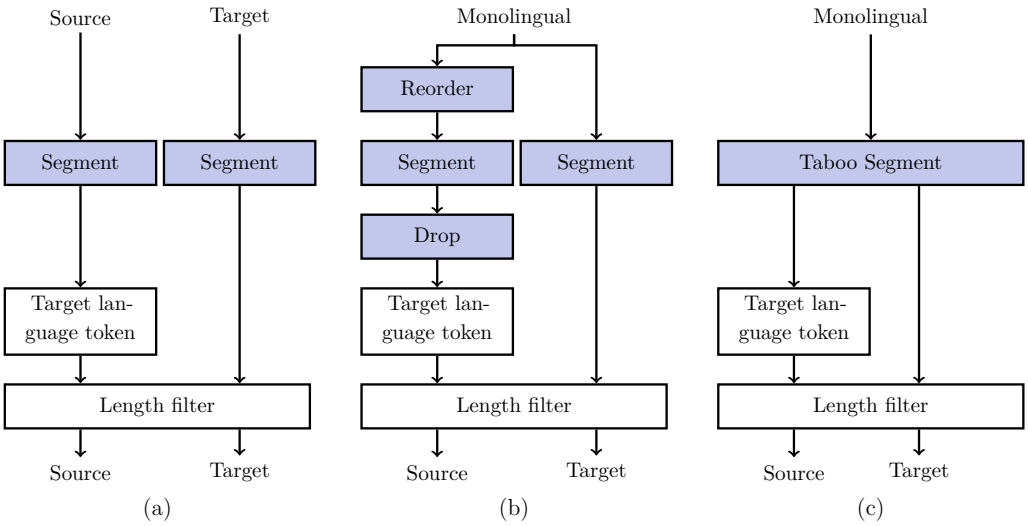


Figure 5.14. Transformations applied to data at training time. Steps with blue background are part of the stochastic noise model. Steps with white background are the deterministic target language token prefixing and length filtering. Length filtering must be applied after segmentation, which may make the sequence longer.

Token masking. Masked language models (Devlin et al., 2019; Song et al., 2019; Lewis et al., 2019; Joshi et al., 2020) apply a special case of token substitution, randomly substituting tokens or spans of tokens with a mask symbol.

Word boundary noise is proposed in Publication IX. In another special case of token substitution, the substituted token is selected deterministically as the token with a word boundary marker either added or removed. E.g. *"kielinen"* would be substituted by *"▁kielinen"* and vice versa. This noise is aimed at the same issue as the boundary correction procedure (Section 5.5.5), e.g. improving robustness to compounding mistakes such as *"*suomen kielinen"* (Finnish speaker).

Subword regularization is a technique proposed by Kudo (2018) to harness the segmentation ambiguity as a source of noise to improve robustness. While most segmentation methods aim to limit the segmentation ambiguity as much as possible, a probabilistic subword segmentation model can be used to generate more variability in the input text. Each time a word token is used during training, a new segmentation is sampled for it. Subword regularization is inspired by latent sequence decompositions (Chan et al., 2016). The methods can be seen as treating the subword segmentation as a latent variable. While marginalizing over the latent variable exactly is intractable, the subword regularization procedure approximates it through sampling. When training a neural model, sampling can be performed on the fly for each minibatch.

Taboo sampling. Publication IX introduces a taboo sampling task for improving the modeling of segmentation ambiguity. Taboo sampling segmentation is a special form of subword regularization (Kudo, 2018) for monolingual data. The

method takes a single word sequence as input, and outputs two different segmentations for it, such that the two segmentations consist of different subwords, whenever possible. Only single character morphs are allowed to be reused on the other side, to avoid failure if no alternative exists. E.g. *"unreasonable"* could be segmented into *"un+reasonable"* on the source side and *"unreason+able"* on the target side. When converted into numerical indices into the lexicon, these two representations are completely different. The task aims to teach the model to associate with each other the multiple ambiguous ways to segment a word, by using a segmentation-invariant internal representation.

For each word, one segmentation is sampled in the usual way, after which another segmentation is sampled using taboo sampling. During taboo sampling, all multi-character subwords used in the first segmentation have their emission probability temporarily set to zero. To avoid introducing a bias from having all the taboo sampled segmentations on the same side, the sides are mixed by swapping the source and target segmentations based on a binary mask $\boldsymbol{m}$, drawn uniformly from the set of masks of the same length as the sentence and with half the bits set to 1

$$\boldsymbol{m} \in \left\{ \{0,1\}^J : \sum_{i=1}^{J} m_i = \left\lceil \frac{J}{2} \right\rceil \right\}. \tag{5.9}$$

Proposed noise model combinations. Figure 5.14 shows three pipelines for noise model combination used in Publication IX. The pipeline for parallel data (a) consists of only sampling segmentation. The primary pipeline for monolingual data (b) is a concatenation of multiple noise models: local reordering, segmentation, and token deletion. A secondary pipeline for monolingual data (c) uses taboo segmentation. In all cases the output consists of a pair of source and target sequences. Observe that the transformations are applied in the data loader at training time, not as an off-line preprocessing stage. This allows the noise to be resampled for each parameter update, which is critical when training continues for multiple epochs of a small dataset. As a minor downside, the NMT software needs to be modified to accommodate the heavier data loader, while preprocessing generally requires no modifications to the software.

5.5.12 Synthetic data set for multimodal translation

Publication XI addresses multimodal machine translation (MMT), in particular the image-guided translation task (Elliott et al., 2015; Specia et al., 2016), in which the input consists of an image and an associated source language caption, and the desired output is the target language caption. The task can be viewed as extending textual translation with conditioning on a second input in a complementary modality. One of the major challenges in this task is the small amount of primary training data containing all three parts: image, source text, and target text. There is more abundant auxiliary data from related tasks, in the form of image captioning data (image and source text, but no target text) and text-only

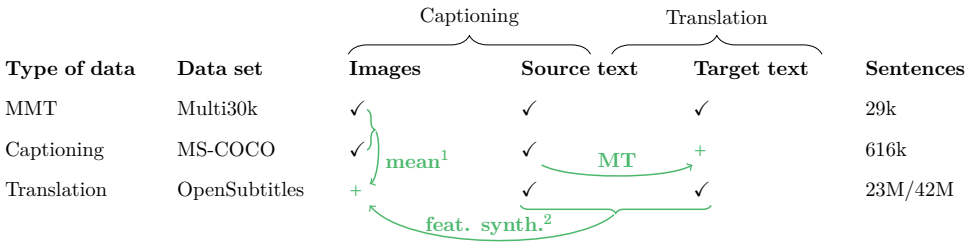


Figure 5.15. Approach for producing synthetic data for multimodal translation. Green plus signs indicate synthetic data. For synthesizing image features, taking the mean of existing image features (1) and using a feature synthesis model (2) are alternative solutions.

translation data (source and target text, but no image). The three types of data are shown in Figure 5.15.

In Publication XI, the auxiliary data was exploited by augmenting it with the missing parts, to create synthetic training triples. To augment image captioning data, the source language captions were "forward-translated" into the target language, with a separate text-only translation system. The downsides with this form of data augmentation are the fact that noisy machine translation output is used on the target side of the synthetic examples, and the inability of the text-only translation system to use the image content. As a result, these synthetic examples cannot teach the system to disambiguate based on the image content, and instead bias the system towards resolving translational ambiguity by selecting the most likely option based on *textual* context.

To augment the text-only translation data, dummy image features need to be generated. In Publication XI, the dummy features were chosen to be the mean of the image features in the training data, indicated by (1) in Figure 5.15. This simple dummy feature is conceptually similar to the back-translation alternative using a single dummy token on the source side (Sennrich et al., 2016).[34] Unfortunately, the dummy features are not informative for resolving translational ambiguity. **(RQ3.5)** The large amount of data using the dummy features biases the model away from using the image features. To explore the effect of the visual features on the translation, an adversarial evaluation was performed, in which the multimodally trained model was deprived of the image features during decoding. Instead of feeding in the actual visual features for the sentence, the mean vector of all visual features in the training data was fed in. This resulted in only small differences to the translated sentences, mostly consisting of minor variations in word order, and nearly no effect on BLEU scores. It is therefore clear that the system does not effectively exploit the visual modality.

Despite these two detrimental biases and the resulting underuse of the visual information, the MeMAD system[35] in Publication XI had the highest performance of the task participants for the English→German and English→French language

[34]See Section 5.3.4 on use of synthetic data.

[35]Software available at `https://github.com/Waino/OpenNMT-py/tree/develop_mmod`.

pairs, with human evaluators significantly preferring the MeMAD system over the other systems (Barrault et al., 2018). In the automatic evaluation there was a margin of +6.0 and +3.5 BLEU respectively, to the other participants. This result indicates that in this particular task, the general translation quality, and in particular fluency of output, was much more important than any successful visual disambiguation.

In spoken language translation, Jia et al. (2019) and Pino et al. (2019) use a similar approach to create synthetic training data. In this task, training triples consist of source audio, source transcript, and target text. A text-only translation system is used to augment automatic speech recognition (ASR) data consisting of source audio and source transcript in the same way as the image captioning data in Publication XI. The text-only translation data is exploited in a different way, however: synthetic speech audio is created using a text-to-speech (TTS) system. This approach suggests a potential alternative feature synthesis method for the image-guided translation task: training a separate feature synthesis model to generate synthetic image features from the text data, indicated by (2) in Figure 5.15. While synthesizing an actual image from the source text would be a challenging and heavy task, it is not necessary to do so, as only the compact features extracted from the image need to be synthesized.

6. Conclusions

> “The real voyage of discovery consists not in seeking new lands but seeing with new eyes.”
> (Marcel Proust, 1923)

This thesis has addressed the task of machine translation into low-resource morphologically rich languages, by focusing on subword vocabulary construction. Developing machine translation (MT) for these languages is important for enabling access to information in a multilingual world that is rapidly becoming more and more digital. There is an opportunity for cross-lingual transfer in multilingual settings, with different languages and language pairs constituting multiple tasks that can benefit from each other. Improvements in the ability to use auxiliary data for training reduce the needed amount of parallel training data, making MT feasible for more low-resource languages.

The alternative setting of fully unsupervised machine translation, while scientifically interesting, is unrealistic in practice (Artetxe et al., 2020b). The asymmetric-resource setting, on the other hand, is very important for practical applications, as small amounts of cross-lingual training signal, and larger amount of auxiliary data are in practice always available. The challenge is learning from a large amount of background data, but very little of exactly the right type of data. In few-shot learning, currently a hot topic in machine learning research, the system must learn from the background data without becoming too strongly biased to prevent adaptation to new tasks. Augmenting with synthetic data raises concerns of distributional mismatch between the natural and synthetic data. When large amounts of synthetic data are added, the aspects that are well represented by the synthetic data may improve, while other aspects deteriorate.[1]

In this thesis, two main approaches were used to address the low-resource condition: improving subword vocabulary construction to make the learning more data efficient, and exploiting auxiliary data via transfer, both in the form of cross-lingual transfer and transfer from monolingual tasks.

A central theme of this thesis was the choice of basic units for representing language. By segmenting words into subwords, it is possible to both get frequent, easily learned representations, and to increase the symmetry between languages. **(RQ1.1)** In low-resource settings and related target languages, optimal vocabularies are small. For deterministic segmentation, such as BPE, the effect is more pronounced. Using regularizing methods such as a subword sampling procedure

[1] See **(RQ3.5)** for an experiment highlighting this challenge.

or a denoising sequence autoencoder (DSAE) lessens the impact of the subword vocabulary size. When training multilingual segmentation models, it is important to use a training data distribution that is balanced between languages, to avoid bias towards the units needed for the high-resource languages.

(RQ1.2) It is not enough to select the size of the vocabulary, as the design of the data-driven segmentation method also affects the distribution of the subwords. Avoiding rare subwords is important. The segmentation can be improved by increasing the consistency, or by embracing the variance in segmentation as a source of noise for regularization. The two aims are only contradictory in extreme cases, and future work could strive to combine them.

(RQ1.3) Learning setups, such as semi-supervised learning and active learning, can improve segmentation consistency and reduce the annotation effort in morphological segmentation. Morfessor FlatCat improves language-internal consistency with hidden Markov model morphotactics. The flat lexicon allows a factorization of the loss function that enables tuning, making Morfessor FlatCat excel at semi-supervised training, with the best performance achieved by using Morfessor FlatCat segmentations as features in a conditional random field. Active learning is superior to random selection for collecting annotations for semi-supervised learning. The best query strategies are the proposed IFSUBSTRINGS and uncertainty sampling combined with representative sampling. Omorfi-restricted Morfessor applies another way to improve language-internal consistency, by using linguistic knowledge from a rule-based morphological analyzer. Even though linguistic morphs are not the optimal granularity for MT, it is not necessary for data-driven segmentation to entirely ignore them.

Cross-lingual consistency can be between source and target or between multiple target languages, and it can focus on overall symmetry between languages or copyable rare items in particular. **(RQ1.4)** In this thesis, cross-lingual transfer was used to improve symmetry in two ways. One approach tunes segmentation towards a matching source–target granularity for SMT, outperforming a word-level baseline. In the other, Cognate Morfessor improves target–target consistency in a one-to-many multilingual setting, by using automatically extracted cognate pairs to connect the segmentations of the two target languages. The cross-lingual segmentation is beneficial for the lower-resourced ENG→EST, outperforming the BPE system. Rare content words are very important for translation adequacy, but unfortunately rare items pose a challenge for machine learning. For a special class of rare words—proper names—copying from source to target is possible. When generating rare words on a character-level in a multi-scale decoder, segmenting the copyable source words improves source–target consistency, and allows an attention mechanism to perform character-by-character copying or transliteration.

As an alternative to seeking maximally consistent segmentations, subword regularization (Kudo, 2018) makes use of segmentation ambiguity as a regularizer, essentially treating the subword segmentation as a latent variable to be approximately marginalized over. Morfessor EM+Prune not only finds models with both lower cost and better quality in unsupervised segmentation than Morfessor

Baseline, but also retains the uncertainty of segmentation by estimating expected occurrence counts rather than assigning all the probability mass to a single segmentation for each word. This makes the method particularly well suited for the sampling of alternative segmentations required by subword regularization. A novel taboo sampling segmentation task creates a source–target pair from monolingual data, by segmenting it in two ways that do not share subword units. Learning to map between the segmentations forces learning of representations that are robust to different segmentations. As a practical recommendation, when using data augmentation techniques such as subword regularization or DSAE, new noised examples should be resampled for each minibatch. This requires MT software with support for stochastic transformations in the data loader.[2] Generating a fixed number of variants in preprocessing is a workaround that results in wasted disk space and inferior quality. Devlin et al. (2019) generated 10 replicas of masked data in preprocessing, but Liu et al. (2019) showed that dynamically sampling the mask for each minibatch is superior.

(RQ2.1) Subword information can improve evaluation of translation into morphologically rich languages, as evidenced by the high correlation with human evaluators of metrics such as LeBLEU and chrF.

Cross-lingual transfer is highly effective in the many-to-one setting, but less so in the one-to-many setting used in this thesis (Arivazhagan et al., 2019b; Kocmi, 2019). However, in very low-resource settings the regularizing effect of the transfer is crucial, making it feasible to train NMT models for extremely low-resource language pairs that lack sufficient resources for the training of standard models. To train an NMT system for translation into a low-resource morphologically rich language in a practical setting, additional resources from related tasks should be exploited.

(RQ3.1) When training only on low-resource tasks, it is beneficial to reduce the model size, train with smaller minibatches, and regularize heavily (Nguyen and Chiang, 2018; Sennrich and Zhang, 2019). However, the benefit from auxiliary multilingual and monolingual data outweighs the improvement from optimizing training for low-resource settings. When auxiliary data is available, using it in large models with large batch sizes performs better.

(RQ3.2) When resources for the translation task of interest are scarce, the most important auxiliary data is parallel data from related languages. Multilingual training improves quality for both low-resource and medium-resource language pairs. The largest individual gains of up to +12.7 BLEU came from cross-lingual transfer. Other types of auxiliary data, such as monolingual corpora, are also beneficial and the gains are partly cumulative.

(RQ3.3) The results of this thesis support the conclusion that language relatedness is more important than parent dataset size. This result is in accordance with Zoph et al. (2016) and Dabre et al. (2017) and in contradiction with Kocmi

[2] A prototype implemented in my dynamicdata fork of OpenNMT-py `https://github.com/Waino/OpenNMT-py`. Later, the dataloader of OpenNMT-py version 2.0 was redesigned to incorporate my proposals.

and Bojar (2018).

(RQ3.4) Both DSAE and back-translation (Sennrich et al., 2016) are effective ways to exploit monolingual data. The DSAE on its own resulted in up to +5.5 BLEU improvement. Back-translation was only effective when combined with the other techniques, but not sufficient on its own. Combining all the techniques resulted in gains of up to +15.6 BLEU. Monolingual auxiliary tasks were most useful for the low-resource target language and the source language, but not for the high-resource target language. A low-resource target-language autoencoder was beneficial even when using multilingual training, but inconclusive together with back-translated data. DSAE should be used when training the model for producing the back-translation.

(RQ3.5) The experiment in multimodal translation is an example of how challenging asymmetric data conditions can be. When large amounts of synthetic data are added, the aspects that are well represented by the synthetic data may improve, while other aspects deteriorate. This effect was shown in multimodal translation, where overall fluency was greatly improved, while simultaneously biasing the system away from using the additional input modality.

(RQ3.6) Combining sequential and parallel transfer results in a scheduled multi-task learning technique for settings with multiple asymmetric-resource tasks. The model using 3-phase scheduled multi-task learning outperforms fully sequential (+2.6 BLEU) and fully parallel transfer (+2.0 BLEU). Fine-tuning only on the low-resource tasks is prone to overfitting. How to optimize weights for the task-mix schedule without resorting to a heavy grid search is still an open question.

(RQ4.1) Although more data is better, 10 000 sentence pairs of parallel data is sufficient to reach above 13 BLEU already with multilingual training and DSAE, but without the use of back-translation. Increasing the amount of data to 18 000 pairs yields diminishing returns of only a 5% relative increase. Adding back-translation brings the quality at 18k pairs to a quite acceptable level of 18.05 BLEU, reaching within a few BLEU points of my previous medium-resource results,[3] while using under 2% of the amount of parallel data.

Considering how much can be accomplished with such small amounts of data, I would like to propose for future work a return to the idea of using elicitation corpora in MT system development (Nirenburg, 1998; Probst and Levin, 2002), but this time in combination with NMT. A small uncontrolled corpus risks being redundant or skewed in domain, which would be likely to hurt performance. An alternative is a designed, language-independent elicitation corpus, covering linguistic phenomena as widely as possible (Beale, 2012; Sylak-Glassman et al., 2016). In addition to containing lexical items for the most important concepts, sets of sentences would be constructed to include features such as number, temporal relationships, aspect, mood, and spatial orientation covering all the values these features take on (such as singular, dual, plural, and paucal for number). As a distinction from what linguists usually do, the corpus would capture also the

[3] As shown in Table 5.3 on page 140.

frequent and ordinary phenomena, instead of concentrating on the distinguishing and exceptional. If commercial translation services are available for the language, the estimated cost of translating such a corpus of 10k sentences would amount to less than 50 000€[4], substantially increasing the number of language pairs for which quality NMT is feasible. The translated corpora would constitute a massively multilingual parallel corpus.

When a natural disaster, epidemic, or other crisis occurs, a low-resource language (LRL) spoken in an affected area may suddenly become highly interesting. There is a need both for translation from the LRL to enable non-natives to monitor local reporting, and also to translate informational material into the LRL. If a massively multilingual universal parent system has been trained ahead-of-time, it can be adapted using transfer learning to rapidly develop a new MT system (Neubig and Hu, 2018; Gheini and May, 2019). Scheduled multi-task learning can still be applied, by first pretraining on a mix of high-resource tasks, and later fine-tuning on a mix including the new target language (the child LRL).

Even though the child LRL does not need to be present in the parallel training data of the universal parent system, the segmentation that the parent was trained on is not easily changed. Word-level models are not well suited to morphologically rich languages, particularly in low-resource settings. Therefore it is important to develop a universal segmentation, that is able to represent all languages of interest equally well. To date, most attempts at universal lexical representations have chosen (whitespace delimited) words as the basic unit (Ammar et al., 2016; Gu et al., 2018a). To account for varying granularity between languages, a truly universal representation should represent concepts or atomic units of meaning, that do not depend on the conventions of a particular language. From an ethical viewpoint, a language that is poorly represented by the chosen standard universal representations would be unable to fully benefit from cross-lingual transfer.

Linguistic annotation is typically produced on a word level, but using word-level annotations in a subword model is not trivial. One method exploiting word-level annotations, using a hybrid word–character model, was presented. Operating on the word level is not equally suitable for all languages, so perhaps more fine-grained annotations on the level of morphemes would be useful for morphologically rich languages.

In the mainstream of neural network research, the ideal is to completely eschew engineered representations, instead allowing a large, deep network to learn implicit representations that are distributed through the intermediary states. In the case of textual NLP, going in this direction means using character-level models. The model must be deep enough so that it can perform the necessary computation to extract and refine the features. Training sequences are long, due to the fine granularity of the input. This line of research involves using enormous amounts of data with a model that makes few assumptions. The computations

[4]Estimated using a price of 0.40€ per word, and sentences of 12.5 words.

incur a substantial cost, both monetary and in the form of energy and associated CO_2 emissions. To train a single BERT (Devlin et al., 2019) model on V100x64 would cost up to $12 571 and emit approximately 652 kg of CO_2, roughly equivalent to a trans-American flight (Strubell et al., 2019). GPT-3 (Brown et al., 2020), an enormous 175 billion parameter model which took hundreds of zetta-operations[5] of computation to train, has been estimated to cost $4.6M to train using a commercial V100 cloud instance (Li, 2020). The cost of training could become a limiting factor in the ability of organizations to perform NLP research. On a positive note, the cost to train a universal parent model can be amortized over multiple instances of adaptation, and the cost of a single adaptation is much more reasonable.

An alternative strategy uses more moderate model sizes and amounts of data, together with sophisticated modeling. Instead of relying on quantity, improvements would be made to the ability of the model to generalize, so that less data is sufficient. In this strategy, selecting suitable subword units for optimal transfer and generalization may still play an important role. It should also not be forgotten that language is symbolic. The ability to compose those symbols and reason about the meanings of the compositions makes language into the powerful tool that it is. New paradigms based on hybridizing neural and symbolic computation (Besold et al., 2017) or modeling compositionality and causality (Lake et al., 2015) could also bring the selecting of basic units to the forefront.

To conclude, the work on this thesis has been an exciting and wondrous journey, that started during the paradigm shift to NMT and followed its spectacular rise for a while. I am happy to have contributed to it in some small way.

[5]Converted from the odd unit petaflop/s-days used by Brown et al. (2020). A better unit for computation is needed.

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL): Volume 2, Short Papers*, pages 242–247.

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884.

Ethem Alpaydin. 2010. *Introduction To Machine Learning.* MIT Press, Cambridge, Massachusetts, USA.

Vamshi Ambati. 2012. *Active learning and crowd-sourcing for machine translation in low-resource scenarios.* PhD Thesis, School of Computer Science, Carnegie Mellon University.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively multilingual word embeddings. ArXiv:1602.01925 [cs.CL].

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Roee Aharoni, Melvin Johnson, and Wolfgang Macherey. 2019a. The missing ingredient in zero-shot neural machine translation. ArXiv:1903.07091 [cs.CL].

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019b. Massively multilingual neural machine translation in the wild: Findings and challenges. ArXiv:1907.05019 [cs.CL].

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Unsupervised statistical machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3632–3642.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2019. An effective approach to unsupervised machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 194–203.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018b. Unsupervised neural machine translation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR).*

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020a. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL).*

Mikel Artetxe, Sebastian Ruder, Dani Yogatama, Gorka Labaka, and Eneko Agirre. 2020b. A call for more rigor in unsupervised cross-lingual learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ehsaneddin Asgari and Hinrich Schütze. 2017. Past, present, future: A computational investigation of the typology of tense in 1000 languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 113–124.

Duygu Ataman and Marcello Federico. 2018. An evaluation of two vocabulary reduction methods for neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (AMTA) (Volume 1: Research Papers)*, pages 97–110.

Duygu Ataman, Matteo Negri, Marco Turchi, and Marcello Federico. 2017. Linguistically motivated vocabulary reduction for neural machine translation from Turkish to English. *The Prague Bulletin of Mathematical Linguistics*, 108(1):331–342.

Automatic Language Processing Advisory Committee. 1966. *Language and Machines: Computers in Translation and Linguistics; A Report.* National Academy of Sciences, National Research Council.

Eleftherios Avramidis and Philipp Koehn. 2008. Enriching morphologically poor languages for statistical machine translation. In *Proceedings of the 2008 Conference of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 763–770.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. ArXiv:1607.06450 [stat.ML].

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING), Volume 1*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Tamali Banerjee and Pushpak Bhattacharyya. 2018. Meaningless yet meaningful: Morphology grounded subword-level NMT. In *Proceedings of the Second Workshop on Subword/Character Level Models*, pages 55–60. Association for Computational Linguistics.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2019. Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies , Volume 1 (Long and Short Papers)*, pages 58–68.

Ankur Bapna, Mia Xu Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. 2018. Training deeper neural machine translation models with transparent attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3028–3033.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1538–1548.

Yehoshua Bar-Hillel. 1959. *Report on the state of machine translation in the United States and Great Britain*. Hebrew University for the United States Office of Naval Research Information.

Marco Baroni, Johannes Matiasek, and Harald Trost. 2002. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning*, pages 48–57, Morristown, NJ, USA. ACL.

Loïc Barrault, Ondřej Bojar, Marta R Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, et al. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (WMT) (Volume 2: Shared Task Papers, Day 1)*, pages 1–61.

Loïc Barrault, Fethi Bougares, Lucia Specia, Chiraag Lala, Desmond Elliott, and Stella Frank. 2018. Findings of the third shared task on multimodal machine translation. In *Proceedings of the Third Conference on Machine Translation (WMT): Shared Task Papers*, pages 304–323.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. 2012. Theano: new features and speed improvements. ArXiv:1211.5590 [cs.SC].

Leonard E. Baum. 1972. An inequality and an associated maximization technique in statistical estimation of probabilistic functions of a Markov process. *Inequalities*, 3(1):1–8.

Stephen Beale. 2012. Documenting endangered languages with linguist's assistant. *Language documentation & conservation*, 6:104–134.

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. ArXiv:1711.02173 [cs.CL].

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 861–872.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2007. Greedy layer-wise training of deep networks. In *Proceedings of Advances in neural information processing systems (NIPS)*, pages 153–160.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 257–267.

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Adam L Berger, Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, John R Gillett, John D Lafferty, Robert L Mercer, Harry Printz, and Luboš Ureš. 1994. The Candide system for machine translation. In *Proceedings of the workshop on Human Language Technology*, pages 157–162. Association for Computational Linguistics.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: A CPU and GPU math compiler in Python. In *Proceedings of the 9th Python in Science Conference*, volume 1, pages 3–10.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *Journal of Artificial Intelligence Research*, 55:409–442.

Delphine Bernhard. 2009. MorphoNet: Exploring the use of community structure for unsupervised morpheme analysis. In *Working notes for the CLEF 2009 Workshop*, Corfu, Greece.

Tarek R. Besold, Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kuehnberger, Luis C. Lamb, Daniel Lowd, Priscila Machado Vieira Lima, Leo de Penning, Gadi Pinkas, Hoifung Poon, and Gerson Zaverucha. 2017. Neural-symbolic learning and reasoning: A survey and interpretation. ArXiv:1711.03902 [cs.AI].

Balthasar Bickel and Johanna Nichols. 2013. Exponence of selected inflectional formatives. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multitask learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL): Volume 2, Short Papers*, pages 164–169.

Arianna Bisazza and Marcello Federico. 2009. Morphological pre-processing for Turkish to English statistical machine translation. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*.

Arianna Bisazza and Clara Tump. 2018. The lazy encoder: A fine-grained analysis of the role of morphology in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2871–2876, Brussels, Belgium. Association for Computational Linguistics.

Christopher M. Bishop. 2006. *Pattern recognition and machine learning*. Springer, New York, USA.

Johannes Bjerva. 2017. Will my auxiliary tagging task help? estimating auxiliary tasks effectivity in multi-task learning. In *Proceedings of the 21st Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 216–220.

Graeme Blackwood, Miguel Ballesteros, and Todd Ward. 2018. Multilingual neural machine translation with task-specific attention. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3112–3122.

Michael Bloodgood and Benjamin Strauss. 2017. Using global constraints and reranking to improve cognates detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1983–1992.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. ArXiv:2004.03720 [cs.CL].

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1899–1907.

Hervé Bourlard and Yves Kamp. 1988. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294.

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A statistical approach to machine translation. *Computational linguistics*, 16(2).

Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, John D Lafferty, and Robert L Mercer. 1992. Analysis, statistical transfer, and synthesis in machine translation. In *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 83–100.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Ralf D Brown. 2002. Corpus-driven splitting of compound words. In *Proceedings of the 9th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 12–21.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. ArXiv:2005.14165 [cs.CL].

Burcu Can and Suresh Manandhar. 2009. Unsupervised learning of morphology by using syntactic categories. In *Working Notes, CLEF 2009 Workshop*, page 10.

Burcu Can and Suresh Manandhar. 2012. Probabilistic hierarchical clustering of morphological paradigms. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 654–663.

Fabienne Cap, Alexander Fraser, Marion Weller, and Aoife Cahill. 2014. How to produce unseen teddy bears: Improved morphological processing of compounds in SMT. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 579–587, Gothenburg, Sweden. Association for Computational Linguistics.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Isaac Caswell, Ciprian Chelba, and David Grangier. 2019. Tagged back-translation. In *Proceedings of the Fourth Conference on Machine Translation (WMT) (Volume 1: Research Papers)*, pages 53–63.

Augustin Cauchy. 1847. Méthode générale pour la résolution des systemes d'équations simultanées. *Compte rendu des séances de l'académie des sciences*, 25(1847):536–538. Translated by Richard J. Pulskamp (2010).

William Chan, Yu Zhang, Quoc Le, and Navdeep Jaitly. 2016. Latent sequence decompositions. ArXiv:1610.03035 [stat.ML].

Yun Chen, Yang Liu, Yong Cheng, and Victor OK Li. 2017. A teacher-student framework for zero-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 1925–1935.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 794–803.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 1965–1974.

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4295–4305, Brussels, Belgium. Association for Computational Linguistics.

Rohan Chitnis and John DeNero. 2015. Variable-length word encodings for neural translation models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2088–2093.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar.

Lonnie Chrisman. 1991. Learning recursive distributed representations for holistic computation. *Connection Science*, 3(4):345–366.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 2: Short Papers)*, pages 385–391, Vancouver, Canada. Association for Computational Linguistics.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical multiscale recurrent neural networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 1693–1703.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proceedings of the NIPS 2014 Workshop on Deep Learning*.

Tagyoung Chung and Daniel Gildea. 2009. Unsupervised tokenization for machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 2, pages 718–726, Singapore. Association for Computational Linguistics.

Alina Maria Ciobanu and Liviu P. Dinu. 2014. Automatic detection of cognates using orthographic alignment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 2, pages 99–105. Association for Computational Linguistics.

Ann Clifton and Anoop Sarkar. 2011. Combining morpheme-based machine translation with post-processing morpheme prediction. In *Proceedings of the 49th Annual Meetong of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, page 11.

Adam Coates, Brody Huval, Tao Wang, David Wu, Bryan Catanzaro, and Ng Andrew. 2013. Deep learning with cots hpc systems. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1337–1345.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Alexis Conneau and Guillaume Lample. 2019a. Cross-lingual language model pretraining. In *Proceedings of Advances in neural information processing systems (NIPS)*, pages 7059–7069.

Alexis Conneau and Guillaume Lample. 2019b. Cross-lingual language model pretraining. In *Proceedings of Advances in neural information processing systems (NIPS)*, pages 7059–7069.

Marta R. Costa-jussà, Carlos Escolano, and José A. R. Fonollosa. 2017. Byte-based neural machine translation. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 154–158, Copenhagen, Denmark. Association for Computational Linguistics.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 2: Short Papers)*, pages 357–361, Berlin, Germany. Association for Computational Linguistics.

Marta R Costa-jussà, Marcos Zampieri, and Santanu Pal. 2018. A neural approach to language variety translation. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 275–282.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22.

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015. Labeled morphological segmentation with semi-Markov models. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 164–174, Beijing, China. Association for Computational Linguistics.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2014. Training deep neural networks with low precision multiplications. ArXiv:1412.7024 [cs.LG].

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. 2007. Analysis of morph-based speech recognition and the modeling of out-of-vocabulary words across languages. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 380–387.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning (MPL)*, volume 6, pages 21–30, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2004. Induction of a simple morphology for highly-inflecting languages. In *Proceedins of the 7th Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON)*, pages 43–51, Barcelona, Spain. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2005a. Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR'05)*, pages 106–113, Espoo, Finland. Helsinki University of Technology, Laboratory of Computer and Information Science.

Mathias Creutz and Krista Lagus. 2005b. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. Technical Report A81, Publications in Computer and Information Science, Publications in Computer and Information Science, Helsinki University of Technology.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1).

Anna Currey and Kenneth Heafield. 2019. Zero-resource neural machine translation with monolingual pivot data. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 99–107.

Anna Currey, Antonio Valerio Miceli-Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation (WMT)*, pages 148–156.

Raj Dabre, Kehai Chen, Benjamin Marie, Rui Wang, Atsushi Fujita, Masao Utiyama, and Eiichiro Sumita. 2019. Nict's supervised neural machine translation systems for the wmt19 news translation task. In *Proceedings of the Fourth Conference on Machine Translation (WMT) (Volume 2: Shared Task Papers, Day 1)*, pages 168–174.

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. 2020. A comprehensive survey of multilingual neural machine translation. ArXiv:2001.01115 [cs.CL].

Raj Dabre, Tetsuji Nakagawa, and Hideto Kazawa. 2017. An empirical study of language relatedness for transfer learning in neural machine translation. In *Proceedings of the 31st Pacific Asia Conference on Language, Information and Computation*, pages 282–286.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Proceedings of Advances in neural information processing systems (NIPS)*, pages 3079–3087.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and improving morphological learning in the neural machine translation decoder. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (IJCNLP) (Volume 1: Long Papers)*, pages 142–151.

Sajib Dasgupta and Vincent Ng. 2007. High-performance, language-independent morphological segmentation. In *Proceedings of the annual conference of the North American Chapter of the ACL (NAACL-HLT)*, pages 155–163, Rochester, NY.

Hal Daumé III. 2009. Semi-supervised or semi-unsupervised. In *NAACL Workshop on Semi-supervised Learning for NLP*.

Carl G. de Marcken. 1996. *Unsupervised Language Acquisition*. PhD Thesis, MIT.

Ferdinand de Saussure. 1916. *Course in general linguistics*. Columbia University Press. (Reprinted in 2011).

Tristan Deleu and Yoshua Bengio. 2018. The effects of negative adaptation in model-agnostic meta-learning. ArXiv:1812.02159 [cs.LG].

Sabine Deligne and Frédéric Bimbot. 1995. Language modeling by variable length sequences: Theoretical formulation and evaluation of multigrams. In *Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 169–172. IEEE.

Sabine Deligne and Frédéric Bimbot. 1997. Inference of variable-length linguistic and acoustic units by multigrams. *Speech Communication*, 23(3):223–241.

Vera Demberg. 2007. A language-independent unsupervised model for morphological segmentation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 45, page 920.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38.

Michael Denkowski and Alon Lavie. 2011. METEOR 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91, Edinburgh, Scotland. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional Transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 1370–1380. Association for Computational Linguistics.

Mattia Antonino Di Gangi and Marcello Federico. 2017. Monolingual embeddings for low resourced neural machine translation. In *Proceedings of the 14th International Workshop on Spoken Language Translation (IWSLT'17)*, pages 97–104.

Anna-Maria Di Sciullo and Edwin Williams. 1987. *On the definition of word*, volume 14 of *Linguistic Inquiry Monographs*. Springer.

Shuoyang Ding, Adithya Renduchintala, and Kevin Duh. 2019. A call for prudent choice of subword merge operations in neural machine translation. In *Proceedings of the XVII Machine Translation Summit*, page 204.

María Do Campo Bayón and Pilar Sánchez-Gijón. 2019. Evaluating machine translation in a low-resource language combination: Spanish-galician. In *Proceedings of Machine Translation Summit XVII Volume 2: Translator, Project and User Tracks*, pages 30–35.

Tobias Domhan and Felix Hieber. 2017. Using target-side monolingual data for neural machine translation through multi-task learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1500–1505.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1723–1732. Association for Computational Linguistics.

Qing Dou and Kevin Knight. 2012. Large scale decipherment for out-of-domain machine translation. In *Proceedings of the 2012 joint conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 266–275.

Qing Dou and Kevin Knight. 2013. Dependency-based decipherment for resource-limited machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1668–1676.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a Dirichlet process mixture model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 616–627. Association for Computational Linguistics.

Gregory Druck, Burr Settles, and Andrew McCallum. 2009. Active learning by labeling features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, volume 1, pages 81–90. Association for Computational Linguistics.

Matthew S. Dryer. 2013. Order of subject, object and verb. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP) (Volume 1: Long Papers)*, pages 334–343.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation:. Technical report, Defense Technical Information Center, Fort Belvoir, VA.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 489–500.

Steffen Eger. 2013. Sequence segmentation by enumeration: An exploration. *The Prague Bulletin of Mathematical Linguistics*, 100(1):113–131.

İlknur Durgar El-Kahlout and Kemal Oflazer. 2006. Initial explorations in English to Turkish statistical machine translation. In *Proceedings of the Workshop on Statistical Machine Translation (WMT)*, pages 7–14. Association for Computational Linguistics.

Desmond Elliott, Stella Frank, and Eva Hasler. 2015. Multi-language image description with neural sequence models. ArXiv:1510.04709 [cs.CL].

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 2: Short Papers)*, pages 72–78.

Carlos Escolano, Marta R Costa-jussà, and José AR Fonollosa. 2019a. From bilingual to multilingual neural machine translation by incremental training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 236–242.

Carlos Escolano, Marta R Costa-jussà, and José AR Fonollosa. 2019b. Towards interlingua neural machine translation. ArXiv:1905.06831 [cs.CL].

Ramy Eskander, Judith L Klavans, and Smaranda Muresan. 2019. Unsupervised morphological segmentation for low-resource polysynthetic languages. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 189–195.

Ramy Eskander, Owen Rambow, and Tianchun Yang. 2016. Extending the use of adaptor grammars for unsupervised morphological segmentation of unseen languages. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING): Technical Papers*, pages 900–910.

Cristina Espana-Bonet, Adám Csaba Varga, Alberto Barrón-Cedeno, and Josef van Genabith. 2017. An empirical analysis of NMT-derived interlingual embeddings and their use in parallel sentence identification. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1340–1350.

Alex Fink and Sai. 2002. Unker non-linear writing system. `https://s.ai/nlws/` [retrieved 4.6.2020].

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1126–1135. JMLR.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–277.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Mark Fishel and Harri Kirik. 2010. Linguistically motivated unsupervised segmentation for machine translation. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, page 5, Valletta, Malta.

Fabienne Fritzinger and Alexander Fraser. 2010. How to avoid burning ducks: Combining linguistic analysis and corpus statistics for German compound processing. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics (MATR)*, page 11.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Matthias Gallé. 2019. Investigating the effectiveness of bpe: The power of shorter sequences. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1375–1381.

Mercedes García-Martínez, Walid Aransa, Fethi Bougares, and Loïc Barrault. 2020. Addressing data sparsity for neural machine translation between morphologically rich languages. *Machine Translation*, pages 1–20.

Mercedes García-Martínez, Loıc Barrault, and Fethi Bougares. 2016. Factored neural machine translation architectures. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, volume 16, page 8, Seattle, USA.

Mozhdeh Gheini and Jonathan May. 2019. A universal parent model for low-resource neural machine translation transfer. ArXiv:1909.06516 [cs.CL].

Adria de Gispert, Sami Virpioja, Mikko Kurimo, and William Byrne. 2009. Minimum Bayes risk combination of translation hypothesis from alternative morphological decompositions. In *Proceedings of the 2009 Annual Conference of the North american Chapter of the association for Computational Linguistics (NAACL)*, Boulder, CO.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

Sharon Goldwater and David McClosky. 2005. Improving statistical MT through morphological analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 676–683. Association for Computational Linguistics.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2014. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*. Citeseer.

Miguel Graça, Yunsu Kim, Julian Schamper, Shahram Khadivi, and Hermann Ney. 2019. Generalizing back-translation in neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 45–52.

Alex Graves. 2013. Generating sequences with recurrent neural networks. ArXiv:1308.0850 [cs.NE].

Alex Graves. 2016. Adaptive computation time for recurrent neural networks. ArXiv:1603.08983 [cs.NE].

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (ICASSP)*, pages 6645–6649. IEEE.

Spence Green and John DeNero. 2012. A class-based agreement model for generating accurately inflected translations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 146–155. Association for Computational Linguistics.

Annette M.B. de Groot, Lucia Dannenburg, and Janet G. van Hell. 1994. Forward and backward word translation by bilinguals. *Journal of Memory and Language*, 33(5):600–629.

Stig-Arne Grönroos, Kristiina Jokinen, Katri Hiovain, Mikko Kurimo, and Sami Virpioja. 2015. Low-resource active learning of North Sámi morphological segmentation. In *Proceedings of 1st International Workshop in Computational Linguistics for Uralic Languages*, pages 20–33.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O. K. Li. 2018a. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354.

Jiatao Gu, Yong Wang, Yun Chen, Kyunghyun Cho, and Victor O. K. Li. 2018b. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3622–3631.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation.

Rohit Gupta, Laurent Besacier, Marc Dymetman, and Matthias Gallé. 2019. Character-based NMT with transformer. ArXiv:1911.04997 [cs.CL].

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Isabelle Guyon, Gavin C. Cawley, Gideon Dror, and Vincent Lemaire. 2011. Results of the active learning challenge. In *Active Learning and Experimental Design workshop, In conjunction with AISTATS 2010, Sardinia, Italy, May 16, 2010*, pages 19–45.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The FLORES evaluation datasets for low-resource machine translation: Nepali–English and Sinhala–English. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6100–6113.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2017. Effective strategies in zero-shot neural machine translation. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*.

Nizar Habash and Fatiha Sadat. 2006. Arabic preprocessing schemes for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 49–52, New York, USA. Association for Computational Linguistics.

Margaret A. Hafer and Stephen F. Weiss. 1974. Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10(11-12):371–385.

Harald Hammarström and Lars Borin. 2011. Unsupervised learning of morphology. *Computational Linguistics*, 37(2):309–350.

Christian Hardmeier, Sara Stymne, Jörg Tiedemann, and Joakim Nivre. 2013. Docent: A document-level decoder for phrase-based statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 193–198, Sofia, Bulgaria. Association for Computational Linguistics.

Douglas Harper. 2019. Online etymology dictionary: hamlet. `https://www.etymonline.com/word/hamlet` [retrieved 8.10.2019].

Zellig S Harris. 1955. From phoneme to morpheme. *Language*, 31(2):190–222.

Hany Hassan, A. Aue, C. Chen, V. Chowdhary, J. Clark, C. Federmann, X. Huang, M. Junczys-Dowmunt, W. Lewis, M. Li, S. Liu, T.-Y. Liu, R. Luo, A. Menezes, T. Qin, F. Seide, X. Tan, F. Tian, L. Wu, S. Wu, Y. Xia, D. Zhang, Z. Zhang, and M. Zhou. 2018. Achieving human parity on automatic Chinese to English news translation. ArXiv: 1803.05567 [cs.CL].

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016a. Dual learning for machine translation. In *Proceedings of Advances in neural information processing systems (NIPS)*, pages 820–828.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. 2020. Dynamic programming encoding for subword segmentation in neural machine translation. ArXiv:2005.06606 [cs.CL].

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (GELUs). ArXiv:1606.08415 [cs.LG].

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377.

Teemu Hirsimäki, Mathias Creutz, Vesa Siivola, Mikko Kurimo, Sami Virpioja, and Janne Pylkkönen. 2006. Unlimited vocabulary speech recognition with morph language models applied to Finnish. *Computer Speech and Language*, 20(4):515–541.

Teemu Hirsimäki, Janne Pylkkönen, and Mikko Kurimo. 2009. Importance of high-order n-gram models in morph-based speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4):724–732.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Charles F Hockett. 1954. Two models of grammatical description. *Word*, 10(2-3):210–234.

Liang Huang, Kai Zhao, and Mingbo Ma. 2017. When to finish? optimal beam search for neural text generation (modulo beam size). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2134–2139.

Matthias Huck, Simon Riess, and Alexander Fraser. 2017. Target-side word segmentation strategies for neural machine translation. In *Proceedings of the Second Conference on Machine Translation (WMT)*, pages 56–67. Association for Computational Linguistics.

John Hutchins and Evgenii Lovtskii. 2000. Petr Petrovich Troyanskii (1894–1950): A forgotten pioneer of mechanical translation. *Machine translation*, 15(3):187–221.

Iván Igartua. 2015. From cumulative to separative exponence in inflection: Reversing the morphological cycle. *Language*, 91(3):676–722.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP) (Volume 1: Long Papers)*, pages 1681–1691.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP) (Volume 1: Long Papers)*, pages 1–10.

Ye Jia, Melvin Johnson, Wolfgang Macherey, Ron J Weiss, Yuan Cao, Chung-Cheng Chiu, Naveen Ari, Stella Laurenzo, and Yonghui Wu. 2019. Leveraging weakly supervised data to improve end-to-end speech-to-text translation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7180–7184. IEEE.

Justin M Johnson and Taghi M Khoshgoftaar. 2019. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):27.

Mark Johnson. 2008. Unsupervised word segmentation for Sesotho using adaptor grammars. In *Proceedings of the Tenth Meeting of ACL Special Interest Group on Computational Morphology and Phonology*, pages 20–27. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Marcin Junczys-Dowmunt. 2019. Microsoft translator at WMT 2019: Towards large-scale document-level neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (WMT) (Volume 2: Shared Task Papers, Day 1)*, pages 225–233.

Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to remember rare events. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1700–1709.

Zhuoliang Kang, Kristen Grauman, and Fei Sha. 2011. Learning with whom to share in multi-task feature learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 2(3), page 4.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2016. Neural morphological analysis: Encoding-decoding canonical segments. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 961–967. Association for Computational Linguistics.

Katharina Kann, Manuel Mager, Ivan Meza-Ruiz, and Hinrich Schütze. 2018. Fortification of neural morphological segmentation models for polysynthetic minimal-resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 47–57.

Alina Karakanta, Jon Dehdari, and Josef van Genabith. 2018. Neural machine translation for low-resource languages without parallel corpora. *Machine Translation*, 32(1-2):167–189.

Dimitar Kazakov and Suresh Manandhar. 2001. Unsupervised learning of word segmentation rules with genetic algorithms and inductive logic programming. *Machine Learning*, 43(1-2):121–162.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491.

Samarth Keshava and Emily Pitler. 2006. A simpler, intuitive approach to morpheme induction. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 31–35.

Yunsu Kim, Miguel Graça, and Hermann Ney. 2020. When and why is unsupervised neural machine translation useless? ArXiv:2004.10581 [cs.CL].

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*, San Diego, USA.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association for Computational Linguistics*, 6:225–240.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184. IEEE.

Tom Kocmi. 2019. *Exploring Benefits of Transfer Learning in Neural Machine Translation*. PhD Thesis, Charles University.

Tom Kocmi and Ondřej Bojar. 2018. Trivial transfer learning for low-resource neural machine translation. In *Proceedings of the Third Conference on Machine Translation (WMT): Research Papers*, pages 244–252.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *EMNLP-CoNLL*, pages 868–876.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics (EACL)*, volume 1, pages 187–193. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133.

Philipp Koehn, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, Evan Herbst, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, and Christine Moran. 2007. Moses: open source toolkit for statistical machine translation. In *45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Oskar Kohonen, Sami Virpioja, and Mikaela Klami. 2008. Allomorfessor: Towards unsupervised morpheme analysis. In *Working notes for the CLEF 2008 Workshop*, Aarhus, Denmark.

Oskar Kohonen, Sami Virpioja, and Krista Lagus. 2010. Semi-supervised learning of concatenative morphology. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 78–86, Uppsala, Sweden. Association for Computational Linguistics.

Daphne Koller, Nir Friedman, Sašo Džeroski, Charles Sutton, Andrew McCallum, Avi Pfeffer, Pieter Abbeel, Ming-Fai Wong, David Heckerman, Chris Meek, et al. 2007. *Introduction to statistical relational learning*. MIT press.

Grzegorz Kondrak. 2001. Identifying cognates by phonetic and semantic similarity. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics.

Grzegorz Kondrak. 2009. Identification of cognates and recurrent sound correspondences in word lists. *TAL*, 50(2):201–235.

Maarit Koponen, Leena Salmi, and Markku Nikulin. 2019. A product and process analysis of post-editor corrections on neural, statistical and rule-based machine translation output. *Machine Translation*, 33(1-2):61–90.

Kimmo Koskenniemi. 1983. *Two-level morphology: A general computational model for word-form recognition and production*. PhD Thesis, University of Helsinki.

Kimmo Koskenniemi. 2008. How to build an open source morphological parser now. In *Resourceful Language Technology–Festschrift in Honor of Anna Sågvall Hein*, page 86. Uppsala universitet.

Leon G. Kraft. 1949. A device for quantizing, grouping, and coding amplitude modulated pulses. Master's thesis, Electrical Engineering Department, Massachusetts Institute of Technology, Cambridge, MA, USA.

Bartosz Krawczyk. 2016. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232.

Julia Kreutzer and Artem Sokolov. 2018. Learning to segment inputs for NMT favors character-level processing. In *Proceedings of the 15th International Workshop on Spoken Language Translation (IWSLT)*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Proceedings of Advances in neural information processing systems (NIPS)*, pages 1097–1105.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 66–75.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 230–237.

Sneha Kudugunta, Ankur Bapna, Isaac Caswell, and Orhan Firat. 2019. Investigating multilingual nmt representations at scale. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1565–1575.

Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Anoop Kunchukuttan and Pushpak Bhattacharyya. 2016. Orthographic syllable as basic unit for smt between related languages. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1912–1917.

Murathan Kurfalı, Ahmet Üstün, and Burcu Can. 2017. A trie-structured Bayesian model for unsupervised morphological segmentation. In *International Conference on Computational Linguistics and Intelligent Text Processing*, pages 87–98. Springer.

Mikko Kurimo, Seppo Enarvi, Ottokar Tilk, Matti Varjokallio, André Mansikkaniemi, and Tanel Alumäe. 2017. Modeling under-resourced languages for speech recognition. *Language Resources and Evaluation*, 51(4):961–987.

Mikko Kurimo, Ville Turunen, and Matti Varjokallio. 2009. Overview of morpho challenge 2008. In *Evaluating Systems for Multilingual and Multimodal Information Access, 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Aarhus, Denmark, September 17–19, 2008, Revised Selected Papers*, volume 5706 of *Lecture Notes in Computer Science*, pages 951–966. Springer.

Mikko Kurimo, Sami Virpioja, Ville Turunen, and Krista Lagus. 2010a. Morpho challenge 2005-2010: Evaluations and results. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 87–95, Uppsala, Sweden. Association for Computational Linguistics.

Mikko Kurimo, Sami Virpioja, and Ville T. Turunen. 2010b. Overview and results of Morpho Challenge 2010. In *Proceedings of the Morpho Challenge 2010 Workshop*, pages 7–24, Espoo, Finland. Aalto University School of Science and Technology, Department of Information and Computer Science.

John Lafferty, Andrew McCallum, and Fernando C N Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

Krista Lagus, Oskar Kohonen, and Sami Virpioja. 2009. Towards unsupervised learning of constructions from text. In *Proceedings of the Workshop on Extracting and Using Constructions in NLP of 17th Nordic Conference on Computational Linguistics, NODALIDA*, page 6.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Surafel M. Lakew, Matteo Negri, and Marco Turchi. 2020. Low resource neural machine translation: A benchmark for five african languages. ArXiv:2003.14402 [cs.CL].

George Lakoff. 1987. *Women, fire, and dangerous things.* University of Chicago press.

George Lakoff and Mark Johnson. 1980. *Metaphors we live by.* University of Chicago press.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations (ICLR).*

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5039–5049.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations.*

Julia A Lasserre, Christopher M Bishop, and Thomas P Minka. 2006. Principled hybrids of generative and discriminative models. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 87–94. IEEE.

Samuel Läubli, Sheila Castilho, Graham Neubig, Rico Sennrich, Qinlan Shen, and Antonio Toral. 2020. A set of recommendations for assessing human–machine parity in language translation. *Journal of Artificial Intelligence Research*, 67:653–672.

Jean-François Lavallée and Philippe Langlais. 2009. Unsupervised morphological analysis by formal analogy. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 617–624. Springer.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: an automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation - StatMT '07*, pages 228–231, Prague, Czech Republic. Association for Computational Linguistics.

Hai-Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon. 2011. Structured output layer neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5524–5527. IEEE.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. 2011. Modeling syntactic context improves morphological segmentation. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL)*, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

Young-Suk Lee. 2004. Morphological analysis for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 57–60. Association for Computational Linguistics.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Tomer Levinboim. 2017. *Invertibility and Transitivity in Low-Resource Machine Translation*. PhD Thesis, University Of Notre Dame.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. ArXiv:1910.13461 [cs.CL].

Chuan Li. 2020. OpenAI's GPT-3 language model: A technical overview. `https://lambdalabs.com/blog/demystifying-gpt-3/` [retrieved 13.7.2020].

Percy Liang and Dan Klein. 2007. Structured Bayesian nonparametric models with variational inference (tutorial). In *Association for Computational Linguistics (ACL)*.

Jindřich Libovický and Alexander Fraser. 2020. Towards character-level transformer nmt by finetuning subword systems. ArXiv:2004.14280 [cs.CL].

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1520–1530.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. ArXiv:2001.08210 [cs.CL].

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. ArXiv:1907.11692 [cs.CL].

Yichao Lu, Phillip Keung, Faisal Ladhak, Vikas Bhardwaj, Shaonan Zhang, and Jason Sun. 2018. A neural interlingua for multilingual machine translation. In *Proceedings of the Third Conference on Machine Translation (WMT): Research Papers*, pages 84–92.

Jiaming Luo, Karthik Narasimhan, and Regina Barzilay. 2017. Unsupervised learning of morphological forests. *Transactions of the Association for Computational Linguistics*, 5:353–364.

Minh-Thang Luong. 2016. *Neural machine translation*. PhD Thesis, Stanford University.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015a. Multi-task sequence to sequence learning. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Minh-Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP) (Volume 1: Long Papers)*, pages 11–19.

Qingsong Ma, Johnny Wei, Ondřej Bojar, and Yvette Graham. 2019. Results of the wmt19 metrics shared task: Segment-level and strong mt systems pose big challenges. In *Proceedings of the Fourth Conference on Machine Translation (WMT) (Volume 2: Shared Task Papers, Day 1)*, pages 62–90.

Yanjun Ma, Nicolas Stroppa, and Andy Way. 2007. Bootstrapping word alignment via word packing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 304–311.

Dominik Macháček, Jonáš Vidra, and Ondřej Bojar. 2018. Morphological and language-agnostic word segmentation for NMT. In *International Conference on Text, Speech, and Dialogue (TSD)*, pages 277–284. Springer.

Joseph G Makin, David A Moses, and Edward F Chang. 2020. Machine translation of cortical activity to text with an encoder–decoder framework. *Nature Neuroscience*, 23:575–582.

Chaitanya Malaviya, Graham Neubig, and Patrick Littell. 2017. Learning language representations for typology prediction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2529–2535.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5057–5068.

Christopher D Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT press.

Benjamin Marie and Atsushi Fujita. 2018. Unsupervised neural machine translation initialized by unsupervised statistical machine translation. ArXiv:1810.12703 [cs.CL].

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of Advances in neural information processing systems (NIPS)*, pages 6294–6305.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Warren S McCulloch and Walter Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Thomas McFadden. 2003. On morphological case and word-order freedom. In *Annual Meeting of the Berkeley Linguistics Society*, volume 29(1), pages 295–306.

Coşkun Mermer and Ahmet Afşın Akın. 2010. Unsupervised search for the optimal segmentation for statistical machine translation. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 31–36, Uppsala, Sweden. Association for Computational Linguistics.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Ruslan Mitkov, Viktor Pekar, Dimitar Blagoev, and Andrea Mulloni. 2007. Methods for extracting and classifying pairs of cognates and false friends. *Machine Translation*, 21(1):29–53.

Christian Monson, Jaime Carbonell, Alon Lavie, and Lori Levin. 2008. Paramor: Finding paradigms across morphology. In *Advances in Multilingual and MultiModal Information Retrieval, 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007, Budapest, Hungary, September 19-21, 2007, Revised Selected Papers*, Lecture Notes in Computer Science , Vol. 5152. Springer.

Pooya Moradi, Nishant Kambhatla, and Anoop Sarkar. 2019. Interrogating the explanatory power of attention in neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 221–230.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer.

Makoto Morishita, Jun Suzuki, and Masaaki Nagata. 2018. Improving neural machine translation by incorporating hierarchical subword features. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 618–629.

Sjur Moshagen, Tommi A Pirinen, and Trond Trosterud. 2013. Building an open-source development infrastructure for language technology projects. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013)*, pages 343–352.

Aaron Mueller and Yash Kumar Lal. 2019. Sentence-level adaptation for low-resource neural machine translation. In *Proceedings of the 2nd Workshop on Technologies for MT of Low Resource Languages*, pages 39–47.

Aaron Mueller, Garrett Nicolai, Arya D McCarthy, Dylan Lewis, Winston Wu, and David Yarowsky. 2020. An analysis of massively multilingual neural machine translation for low-resource languages. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 3710–3718.

James Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38.

Rudra Murthy, Anoop Kunchukuttan, and Pushpak Bhattacharyya. 2019. Addressing word-order divergence in multilingual neural machine translation for extremely low resource languages. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3868–3873.

Maria Nadejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Birch. 2017. Predicting target language ccg supertags improves neural machine translation. In *Proceedings of the Second Conference on Machine Translation (WMT)*, pages 68–79.

Makoto Nagao. 1984. A framework of a mechanical translation between Japanese and English by analogy principle. *Artificial and human intelligence*, pages 351–354.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Preslav Nakov and Hwee Tou Ng. 2009. Improved statistical machine translation for resource-poor languages using related resource-rich languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1358–1367. Association for Computational Linguistics.

Jason Naradowsky and Sharon Goldwater. 2009. Improving morphology induction by learning spelling rules. In *IJCAI*, pages 1531–1536.

Jason Naradowsky and Kristina Toutanova. 2011. Unsupervised bilingual morpheme segmentation and alignment with context-rich hidden semi-Markov models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 895–904. Association for Computational Linguistics.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2015. An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics*, 3:157–167.

Karthik Narasimhan, Damianos Karakos, Richard Schwartz, Stavros Tsakalidis, and Regina Barzilay. 2014. Morphological segmentation for keyword spotting. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 880–885.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 875–880.

Sylvain Neuvel and Sean A Fulop. 2002. Unsupervised learning of morphology without morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*, pages 31–40. Association for Computational Linguistics.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's wmt19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation (WMT) (Volume 2: Shared Task Papers, Day 1)*, pages 314–319.

Toan Q Nguyen and David Chiang. 2017. Transfer learning across low-resource, related languages for neural machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 296–301.

Toan Q Nguyen and David Chiang. 2018. Improving lexical choice in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 334–343.

Jan Niehues and Eunah Cho. 2017. Exploiting linguistic resources for neural machine translation using multi-task learning. In *Proceedings of the Second Conference on Machine Translation (WMT)*, pages 80–89.

Sonja Nießen and Hermann Ney. 2000. Improving SMT quality with morpho-syntactic analysis. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 1081–1085. Association for Computational Linguistics.

Sergei Nirenburg. 1998. Project boas: "a linguist in the box" as a multi-purpose language resource. In *First International Conference on language resources & evaluation: Granada, Spain, 28-30 May 1998*, pages 739–746. European Language Resources Association.

Kemal Oflazer and Ilknur Durgar El-Kahlout. 2007. Exploring different representational units in English-to-Turkish statistical machine translation. In *Proceedings of the second workshop on statistical machine translation*, pages 25–32. Association for Computational Linguistics.

Aitor Ormazabal, Mikel Artetxe, Gorka Labaka, Aitor Soroa, and Eneko Agirre. 2019. Analyzing the limitations of cross-lingual word embedding mappings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4990–4995.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *40th annual meeting of the association for computational linguistics*, pages 311–318, Philadelphia, PA, USA. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1310–1318.

Peyman Passban. 2018. *Machine Translation of Morphologically Rich Languages Using Deep Neural Networks*. PhD Thesis, School of Computing, Dublin City University.

Peyman Passban, Qun Liu, and Andy Way. 2017. Translating low-resource languages by vocabulary adaptation from close counterparts. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 16(4):1–14.

Peyman Passban, Qun Liu, and Andy Way. 2018. Improving character-based decoding using target-side morphological information for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 58–68.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Barun Patra, Joel Ruben Antony Moniz, Sarthak Garg, Matthew R Gormley, and Graham Neubig. 2019. Bilingual lexicon induction with semi-supervision in non-isometric embedding spaces. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 184–193.

Mārcis Pinnis, Rihards Krišlauks, Daiga Deksne, and Toms Miks. 2017. Neural machine translation for morphologically rich languages with improved sub-word units and synthetic data. In *International Conference on Text, Speech, and Dialogue*, pages 237–245. Springer.

Juan Pino, Liezl Puzon, Jiatao Gu, Xutai Ma, Arya D. McCarthy, and Deepak Gopinath. 2019. Harnessing indirect training data for end-to-end automatic speech translation: Tricks of the trade. In *Proceedings of the 16th International Workshop on Spoken Language Translation (IWSLT)*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4996–5001.

Tommi A Pirinen. 2015. Omorfi—free and open source morphological lexical database for Finnish. In *20th Nordic Conference on Computational Linguistics (NODALIDA)*, Vilnius, Lithuania.

Tommi A Pirinen, Antonio Toral, and Raphael Rubino. 2016. Rule-based and statistical morph segments in English-to-Finnish SMT. In *2nd International Workshop on Computational Linguistics for Uralic Languages*, page 11, Szeged, Hungary.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. 2018. Contextual parameter generation for universal neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–435.

Thierry Poibeau. 2017. *Machine Translation*. The MIT Press, Cambridge, Massachusetts.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 209–217, Boulder, Colorado. Association for Computational Linguistics.

Martin Popel, Dominik Macháček, Michal Auersperger, Ondřej Bojar, and Pavel Pecina. 2019. English-czech systems in wmt19: Document-level transformer. In *Proceedings of the Fourth Conference on Machine Translation (WMT) (Volume 2: Shared Task Papers, Day 1)*, pages 342–348.

Maja Popović. 2017. Comparing language related issues for nmt and pbmt between german and english. *The Prague Bulletin of Mathematical Linguistics*, 108(1):209–220.

Maja Popović, Daniel Stein, and Hermann Ney. 2006. Statistical machine translation of german compound words. In *International Conference on Natural Language Processing (in Finland)*, pages 616–624. Springer.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the 10th Workshop on Statistical Machine Translation (WMT)*, pages 392–395. Association for Computational Linguistics.

Maja Popović. 2016. chrF deconstructed: parameters and n-gram weights. In *Proceedings of the 1st Conference on Machine Translation (WMT)*.

Maja Popović and Hermann Ney. 2004. Improving word alignment quality using morpho-syntactic information. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 310–314. Association for Computational Linguistics.

Lorien Y Pratt, Jack Mostow, and Candace A Kamm. 1991. Direct transfer of learned information among neural networks. In *Proceedings of the ninth National conference on Artificial intelligence-Volume 2*, pages 584–589. AAAI Press.

Princeton University. 2010. About WordNet. `https://wordnet.princeton.edu/` [retrieved 10.4.2019].

Katharina Probst and Lori Levin. 2002. Challenges in automated elicitation of a controlled bilingual corpus. *Proceedings of TMI*.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2019. BPE-Dropout: Simple and effective subword regularization. ArXiv:1910.13267 [cs.CL].

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Proceedings of Advances in neural information processing systems (NIPS)*, pages 6076–6085.

Taraka Rama. 2016. Siamese convolutional networks for cognate identification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1018–1027.

Prajit Ramachandran, Peter J Liu, and Quoc Le. 2017. Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 383–391.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 12–21. Association for Computational Linguistics.

Shuo Ren, Zhirui Zhang, Shujie Liu, Ming Zhou, and Shuai Ma. 2019. Unsupervised neural machine translation with smt as posterior regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 241–248.

Don Ringe. 2017. *From Proto-Indo-European to Proto-Germanic*, volume 1. Oxford University Press.

Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14(5):465–471.

Jorma Rissanen. 1989. *Stochastic Complexity in Statistical Inquiry*, volume 15. World Scientific Series in Computer Science, Singapore.

Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. 2005. To transfer or not to transfer. In *NIPS 2005 workshop on transfer learning*, volume 898, pages 1–4.

Raphael Rubino, Tommi Pirinen, Miquel Esplà-Gomis, Nikola Ljubešić, Sergio Ortiz Rojas, Vassilis Papavassiliou, Prokopis Prokopidis, and Antonio Toral. 2015. Abu-matran at wmt 2015 translation task: Morphological segmentation and web crawling. In *Tenth Workshop on Statistical Machine Translation*, pages 184–191, Lisbon, Portugal. Association for Computational Linguistics.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. ArXiv:1706.05098 [cs.LG].

David E Rumelhart and James L McClelland. 1985. *On learning the past tenses of English verbs*. Institute for Cognitive Science, University of California, San Diego.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and mikko kurimo. 2014. Painless semi-supervised morphological segmentation using conditional random fields. In *European Chapter of the Association for Computational Linguistics (EACL)*, pages 84–89, Gothenburg, Sweden. Association for Computational Linguistics.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 379–389. Association for Computational Linguistics.

Stuart Russell and Peter Norvig. 2010. *Artificial Intelligence – A Modern Approach*. Prentice Hall, Upper Saddle River, New Jersey, USA.

Devendra Singh Sachan and Graham Neubig. 2018. Parameter sharing methods for multilingual self-attentional translation models. In *Proceedings of the Third Conference on Machine Translation (WMT): Research Papers*, pages 261–271.

Tarek Sakakini, Suma Bhat, and Pramod Viswanath. 2017. MORSE: Semantic-ally drive-n MORpheme SEgment-er. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 552–561.

Mohammad Salameh, Colin Cherry, and Grzegorz Kondrak. 2014. Lattice desegmentation for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 100–110. Association for Computational Linguistics.

Mohammad Salameh, Colin Cherry, and Grzegorz Kondrak. 2015. What matters most in morphologically segmented SMT models? *Syntax, Semantics and Structure in Statistical Translation*, page 65.

Elizabeth Salesky, Andrew Runge, Alex Coda, Jan Niehues, and Graham Neubig. 2020. Optimizing segmentation granularity for neural machine translation. *Machine Translation*, pages 1–19.

Sametinget. 2004. Den samiske tekstbanken.

Patrick Schone and Daniel Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–9. Association for Computational Linguistics.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and Korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

Holger Schwenk, Daniel Dchelotte, and Jean-Luc Gauvain. 2006. Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 723–730. Association for Computational Linguistics.

Holger Schwenk and Matthijs Douze. 2017. Learning joint multilingual sentence representations with neural machine translation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 157–167.

Steven L Scott. 2002. Bayesian methods for hidden Markov models: Recursive computing in the 21st century. *Journal of the American Statistical Association*, 97(457):337–351.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The university of edinburgh's neural MT systems for WMT17. In *Proceedings of the Second Conference on Machine Translation (WMT)*, pages 389–399. Association for Computational Linguistics.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the 1st Conference on Machine Translation (WMT)*, pages 83–91. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221.

Siriwan Sereewattana. 2003. *Unsupervised segmentation for statistical machine translation.* Masters Thesis, School of Informatics, University of Edinburgh.

Burr Settles. 2010. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin, Madison.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423.

Aditya Siddhant, Melvin Johnson, Henry Tsai, Naveen Ari, Jason Riesa, Ankur Bapna, Orhan Firat, and Karthik Raman. 2020. Evaluating the cross-lingual effectiveness of massively multilingual neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8854–8861.

Miikka Silfverberg and Mans Hulden. 2018. An encoder-decoder approach to the paradigm cell filling problem. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2883–2889, Brussels, Belgium. Association for Computational Linguistics.

Miikka Silfverberg, Teemu Ruokolainen, Krister Lindén, and Mikko Kurimo. 2016. FinnPos: an open-source morphological tagging and lemmatization toolkit for Finnish. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, volume 50(4), pages 863–878.

Gary F. Simons and Charles D. Fennig, editors. 2018. *Ethnologue: Languages of the World*, twenty-first edition. Dallas, Texas: SIL International. Online version: `http://www.ethnologue.com`. Referenced 12.6.2018.

Mittul Singh, Sami Virpioja, Peter Smit, and Mikko Kurimo. 2019. Subword RNNLM approximations for out-of-vocabulary keyword search. In *Proceedings of the Interspeech 2019*, pages 4235–4239.

Kairit Sirts and Sharon Goldwater. 2013. Minimally-supervised morphological segmentation using adaptor grammars. *Transactions of the Association for Computational Linguistics*, 1:255–266.

Vladimír Skalička. 1979. Das erscheinungsbild der sprachtypen. In *Typologische Studien*, pages 21–58. Springer.

Ivan Skorokhodov, Anton Rykachevskiy, Dmitry Emelyanenko, Sergey Slotin, and Anton Ponkratov. 2018. Semi-supervised neural machine translation with language models. In *Proceedings of the AMTA 2018 Workshop on Technologies for MT of Low Resource Languages (LoResMT 2018)*, pages 37–44.

Peter Smit, Sami Virpioja, Mikko Kurimo, et al. 2017. Improved subword modeling for WFST-based speech recognition. In *INTERSPEECH 2017–18th Annual Conference of the International Speech Communication Association.*

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Matthew Snover, Gaja E Jarosz, and Michael R Brent. 2002. Unsupervised learning of morphology using a novel directed search algorithm: Taking the first step. In *ACL-02 workshop on Morphological and phonological learning*, volume 6, pages 11–20. Association for Computational Linguistics.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of ACL-08: HLT*, pages 737–745, Columbus, Ohio. Association for Computational Linguistics.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 778–788.

Le Hai Son, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of the 2012 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 39–48. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 5926–5936.

Alexey Sorokin. 2019. Convolutional neural networks for low-resource morpheme segmentation: baseline or state-of-the-art? In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 154–159.

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *Proceedings of the First Conference on Machine Translation (WMT): Volume 2, Shared Task Papers*, pages 543–553.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural lattice-to-sequence models for uncertain inputs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1380–1389.

Sebastian Spiegler and Peter A. Flach. 2010. Enhanced word decomposition by calibrating the decision threshold of probabilistic models and using a model ensemble. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 375–383, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sebastian Spiegler and Christian Monson. 2010. EMMA: A novel evaluation metric for morphological analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1029–1037. Association for Computational Linguistics.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011. Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1269–1280. Association for Computational Linguistics.

Tejas Srinivasan, Ramon Sanabria, and Florian Metze. 2019. Multitask learning for different subword segmentations in neural machine translation. ArXiv:1910.12368 [cs.CL].

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. 2017. Cold fusion: Training seq2seq models together with language models. In *Proceedings of the Interspeech 2018*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proceedings of Advances in neural information processing systems (NIPS)*, pages 2377–2385.

Felix Stahlberg, James Cross, and Veselin Stoyanov. 2018. Simple fusion: Return of the language model. In *Proceedings of the Third Conference on Machine Translation (WMT): Research Papers*, pages 204–211.

Miloš Stanojević, Amir Kamran, Philipp Koehn, and Ondřej Bojar. 2015. Results of the WMT15 metrics shared task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 256–273.

Mark Steedman. 2008. Last words–on becoming a discipline. *Computational Linguistics*, 34(1):137–144.

Dave Steinkraus, Ian Buck, and PY Simard. 2005. Using gpus for machine learning algorithms. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 1115–1120. IEEE.

Pavol Štekauer. 2002. On the theory of neologisms and nonce-formations. *Australian Journal of Linguistics*, 22(1):97–112.

Mihaela C Stoian, Sameer Bansal, and Sharon Goldwater. 2020. Analyzing ASR pretraining for low-resource speech-to-text translation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7909–7913. IEEE.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650.

Sara Stymne and Nicola Cancedda. 2011. Productive generation of compound words in statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 250–260, Edinburgh, UK. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of Advances in neural information processing systems (NIPS)*, pages 3104–3112.

John Sylak-Glassman, Christo Kirov, and David Yarowsky. 2016. Remote elicitation of inflectional paradigms to seed morphological analysis in low-resource languages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3116–3120.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

David Talbot and Miles Osborne. 2006. Modelling lexical redundancy for machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 969–976. Association for Computational Linguistics.

Aleš Tamchyna, Marion Weller-Di Marco, and Alexander Fraser. 2017. Modeling target-side inflection in neural machine translation. In *Proceedings of the Second Conference on Machine Translation (WMT)*, pages 32–42. Association for Computational Linguistics.

Brian Thompson, Huda Khayrallah, Antonios Anastasopoulos, Arya D McCarthy, Kevin Duh, Rebecca Marvin, Paul McNamee, Jeremy Gwinnup, Tim Anderson, and Philipp Koehn. 2018. Freezing subnetworks to analyze domain adaptation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation (WMT): Research Papers*, pages 124–132.

Sebastian Thrun. 1995. Lifelong learning: A case study. Technical report, Carnegie-Mellon University, Dept of Computer Science, Pittsburgh, PA, USA.

Jörg Tiedemann. 2009. Character-based PSMT for closely related languages. In *Proceedings of the 13th Conference of the European Association for Machine Translation (EAMT 2009)*, pages 12–19.

Jörg Tiedemann and Yves Scherrer. 2017. Neural machine translation with extended context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 82–92.

Jörg Tiedemann et al. 2018. Emerging language spaces learned from massively multilingual corpora. In *Proceedings of the Digital Humanities in the Nordic Countries 3rd Conference (DHN 2018)*. CEUR Workshop Proceedings.

Andrei N Tikhonov. 1963. Solution of incorrectly formulated problems and the regularization method. *Soviet Mathematics*, 4:1035–1038.

Antonio Toral, Sheila Castilho, Ke Hu, and Andy Way. 2018. Attaining the unattainable? reassessing claims of human parity in neural machine translation. In *Proceedings of the Third Conference on Machine Translation (WMT): Research Papers*, pages 113–123.

Antonio Toral, Lukas Edman, Galiya Yeshmagambetova, and Jennifer Spenader. 2019. Neural machine translation for English–Kazakh with morphological segmentation and synthetic data. In *Proceedings of the Fourth Conference on Machine Translation (WMT) (Volume 2: Shared Task Papers, Day 1)*, pages 386–392.

Antonio Toral and Víctor M. Sánchez-Cartagena. 2017. A multifaceted evaluation of neural versus phrase-based machine translation for 9 language directions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL): Volume 1, Long Papers*, pages 1063–1073.

Antonio Toral and Andy Way. 2018. What level of quality can neural machine translation attain on literary text? ArXiv:1801.04962 [cs].

Lisa Torrey and Jude Shavlik. 2009. Transfer learning. In Emilio Soria Olivas, editor, *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques: Algorithms, Methods, and Techniques*, pages 242–264. IGI Global.

Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. 2008. Applying morphology generation models to machine translation. In *Proceedings of ACL-08: HLT*, pages 514–522.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85.

Alan M. Turing. 1950. Computing machinery and intelligence. *Mind*, LIX(236):433–460.

Ville T. Turunen and Mikko Kurimo. 2011. Speech retrieval from unsegmented Finnish audio using statistical morpheme-like units for segmentation, recognition, and retrieval. *ACM Transactions on Speech and Language Processing*, 8(1):1–25.

Universal Dependencies contributors. 2017. UD v2 guidelines: Tokenization and word segmentation. `https://universaldependencies.org/u/overview/tokenization.html` [retrieved 8.10.2019].

Universal Dependencies contributors. 2020. Universal Dependencies v2.6. `http://hdl.handle.net/11234/1-3226` [retrieved 27.5.2020].

Vaibhav Vaibhav, Sumeet Singh, Craig Stewart, and Graham Neubig. 2019. Improving robustness of machine translation with synthetic noise. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1916–1920.

Aidar Valeev, Ilshat Gibadullin, Albina Khusainova, and Adil Khan. 2019. Application of low-resource machine translation techniques to russian-tatar language pair. ArXiv:1910.00368 [cs.CL].

Matti Varjokallio, Mikko Kurimo, and Sami Virpioja. 2013. Learning a subword vocabulary based on unigram likelihood. In *Proceedings of the 2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 7–12, Olomouc, Czech Republic. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning (ICML)*, pages 1096–1103.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Proceedings of Advances in neural information processing systems (NIPS)*, pages 3630–3638.

Sami Virpioja. 2012. *Learning Constructions of Natural Language: Statistical Models and Evaluations*. PhD Thesis, Aalto University, Espoo, Finland.

Sami Virpioja and Oskar Kohonen. 2009. Unsupervised morpheme analysis with Allomorfessor. In *Working notes for the CLEF 2009 Workshop*, Corfu, Greece.

Sami Virpioja, Oskar Kohonen, and Krista Lagus. 2011a. Evaluating the effect of word frequencies in a probabilistic generative model of morphology. In *Proceedings of the 18th Nordic Conference of Computational Linguistics (NODALIDA 2011)*, volume 11 of *NEALT Proceedings Series*, pages 230–237, Riga, Latvia. Northern European Association for Language Technology.

Sami Virpioja, André Mansikkaniemi, Jaakko Väyrynen, and Mikko Kurimo. 2010. Applying morphological decompositions to statistical machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 201–206. Association for Computational Linguistics.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. 2013. Morfessor 2.0: Python implementation and extensions for Morfessor Baseline. Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, Department of Signal Processing and Acoustics, Aalto University, Helsinki, Finland.

Sami Virpioja, Ville T Turunen, Sebastian Spiegler, Oskar Kohonen, and Mikko Kurimo. 2011b. Empirical comparison of evaluation methods for unsupervised learning of morphology. *Traitement Automatique des Langues*, 52(2):45–90.

Sami Virpioja, Jaakko J Väyrynen, Mathias Creutz, and Markus Sadeniemi. 2007. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. In *Machine Translation Summit XI*, volume 2007, pages 491–498, Copenhagen, Denmark.

A. J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808.

Ekaterina Vylomova, Trevor Cohn, Xuanli He, and Gholamreza Haffari. 2017. Word representation models for morphologically rich languages in neural machine translation. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 103–108.

Raúl Vázquez, Alessandro Raganato, Jörg Tiedemann, and Mathias Creutz. 2019. Multilingual NMT with a language-independent attention bridge. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 33–39.

Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. 1989. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339.

Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. 2016. Morphological segmentation with window LSTM neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2842–2848.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. Switchout: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 856–861.

Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*.

Warren Weaver. 1955. Translation. *Machine translation of languages*, 14:15–23.

Joseph Weizenbaum. 1966. ELIZA – a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Richard Wicentowski. 2004. Multilingual noise-robust supervised morphological analysis using the wordframe model. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology: Current Themes in Computational Phonology and Morphology*, pages 70–77. Association for Computational Linguistics.

Søren Wichmann, Eric W. Holman, and Cecil H. Brown (eds.). 2020. The automatic similarity judgement program database (version 19). `https://asjp.clld.org/` [retrieved 25.5.2020].

Bernard Widrow and Marcian E Hoff. 1960. Adaptive switching circuits. Technical report, Stanford Univ Ca Stanford Electronics Labs.

Shijie Wu, Alexis Conneau, Haoran Li, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Emerging cross-lingual structure in pretrained language models. ArXiv:1911.01464 [cs.CL].

Yingting Wu and Hai Zhao. 2018. Finding better subword segmentation for neural machine translation. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 53–64. Springer.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. ArXiv:1609.08144 [cs].

Zhao Xu, Kai Yu, Volker Tresp, Xiaowei Xu, and Jizhi Wang. 2003. Representative sampling for text classification using support vector machines. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Fabrizio Sebastiani, editors, *Advances in Information Retrieval*, volume 2633 of *Lecture Notes in Computer Science*, pages 393–407. Springer Berlin Heidelberg, Berlin, Heidelberg.

Mei Yang and Katrin Kirchhoff. 2006. Phrase based backoff models for machine translation of highly inflected languages. In *11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised neural machine translation with weight sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 46–55.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 207–216. Association for Computational Linguistics.

Reyyan Yeniterzi and Kemal Oflazer. 2010. Syntax-to-morphology mapping in factored phrase-based statistical machine translation from English to Turkish. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, page 11.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2972–2978.

Poorya Zaremoodi, Wray Buntine, and Gholamreza Haffari. 2018. Adaptive knowledge sharing in multi-task learning: Improving low-resource neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 656–661.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1535–1545.

Pei Zhang, Niyu Ge, Boxing Chen, and Kai Fan. 2019. Lattice transformer for speech translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6475–6484.

Xiaoheng Zhang. 1998. Dialect MT: a case study between Cantonese and Mandarin. In *COLING 1998 Volume 2: The 17th International Conference on Computational Linguistics*.

Xiaojin Zhu. 2006. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2:60.

George Kingsley Zipf. 1932. *Selective Studies and the Principle of Relative Frequency in Language*. Harvard University Press, Cambridge, MA.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568–1575.

Robert Östling, Yves Scherrer, Jörg Tiedemann, Gongbo Tang, and Tommi Nieminen. 2017. The Helsinki neural machine translation system. In *Proceedings of the Second Conference on Machine Translation*, pages 338–347.

Robert Östling and Jörg Tiedemann. 2017a. Continuous multilinguality with language vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL): Volume 2, Short Papers*, pages 644–649.

Robert Östling and Jörg Tiedemann. 2017b. Neural machine translation for low-resource languages. ArXiv:1708.05729 [cs.CL].

Errata

> “Morphology is inherently messy.
> (Joan B. Hooper, 1979)”

Publication III

In Section 1 paragraph 4, Morphological surface segmentation is a related, simplified problem compared to morphological analysis. However, it is not a relaxation but rather a more constrained problem.

Table 1 Only mentions the original size of the evaluation pool (800 word types), but does not give the size of the final annotated test set (796 word types).

Publication VII

In Section 1 paragraph 5, “need to marginalize over the whole vocabulary during prediction” should instead read “the softmax to normalize the output probability distribution requiring a summation over the whole vocabulary”.

In the last paragraph of Section 2.3, the linear combination should be a log-linear combination.

Publication X

The beam search scoring function (Equation 3) adds length normalization lp(y), instead of dividing by it. A corrected equation is shown in Equation 5.8 of the thesis introduction.

Publication I

Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, Dublin, Ireland, pages 1177–1185, Aug 2014.

Morfessor FlatCat: An HMM-Based Method for Unsupervised and Semi-Supervised Learning of Morphology

Stig-Arne Grönroos[1]
stig-arne.gronroos@aalto.fi

Sami Virpioja[2]
sami.virpioja@aalto.fi

Peter Smit[1]
peter.smit@aalto.fi

Mikko Kurimo[1]
mikko.kurimo@aalto.fi

[1]Department of Signal Processing and Acoustics, Aalto University
[2]Department of Information and Computer Science, Aalto University

Abstract

Morfessor is a family of methods for learning morphological segmentations of words based on unannotated data. We introduce a new variant of Morfessor, FlatCat, that applies a hidden Markov model structure. It builds on previous work on Morfessor, sharing model components with the popular Morfessor Baseline and Categories-MAP variants. Our experiments show that while unsupervised FlatCat does not reach the accuracy of Categories-MAP, with semi-supervised learning it provides state-of-the-art results in the Morpho Challenge 2010 tasks for English, Finnish, and Turkish.

1 Introduction

Morphological analysis is essential for automatic processing of compounding and highly-inflecting languages, for which the number of unique word forms may be very large. Apart from rule-based analyzers, the task has been approached by machine learning methodology. Especially unsupervised methods that require no linguistic resources have been studied widely (Hammarström and Borin, 2011). Typically these methods focus on morphological segmentation, i.e., finding *morphs*, the surface forms of the morphemes.

For language processing applications, unsupervised learning of morphology can provide decent-quality analyses without resources produced by human experts. However, while morphological analyzers and large annotated corpora may be expensive to obtain, a small amount of linguistic expertise is more easily available. A well-informed native speaker of a language can often identify the different prefixes, stems, and suffixes of words. Then the question is how many annotated words makes a difference. One answer was provided by Kohonen et al. (2010), who showed that already one hundred manually segmented words provide significant improvements to the quality of the output when comparing to a linguistic gold standard.

The semi-supervised approach by Kohonen et al. (2010) was based on Morfessor Baseline, the simplest of the Morfessor methods by Creutz and Lagus (2002; 2007). The statistical model of Morfessor Baseline is simply a categorical distribution of morphs—a unigram model in the terms of statistical language modeling. As the semi-supervised Morfessor Baseline outperformed all unsupervised and semi-supervised methods evaluated in the Morpho Challenge competitions (Kurimo et al., 2010a) so far, the next question is how the approach works for more complex models.

Another popular variant of Morfessor, Categories-MAP (CatMAP) (Creutz and Lagus, 2005), models word formation using a hidden Markov model (HMM). The context-sensitivity of the model improves the precision of the segmentation. For example, it can prevent splitting a single *s*, a common English suffix, from the beginning of a word. Moreover, it can disambiguate between identical morphs that are actually surface forms of different morphemes. Finally, separation of stems and affixes in the output makes it simple to use the method as a stemmer.

In contrast to Morfessor Baseline, the lexicon of CatMAP is *hierarchical*: a morph that is already in the lexicon may be used to encode the forms of other morphs. This has both advantages and drawbacks.

Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 1177–1185, Dublin, Ireland, August 23-29 2014.

One downside is that it mixes the prior and likelihood components of the cost function, so that the semi-supervised approach presented by Kohonen et al. (2010) is not usable.

1.1 Hierarchical versus flat lexicons

From the viewpoint of data compression and following the two-part Minimum Description Length principle (Rissanen, 1978), Morfessor tries to minimize the number of bits needed to encode both the model parameters and the training data. Equivalently, the cost function L can be derived from the Maximum a Posteriori (MAP) estimate:

$$\hat{\theta} = \arg\max_{\theta} \mathrm{P}(\theta \mid \boldsymbol{D}) = \arg\min_{\theta} \big(-\log \mathrm{P}(\theta) - \log \mathrm{P}(\boldsymbol{D} \mid \theta) \big) = \arg\min_{\theta} L(\theta, \boldsymbol{D}), \qquad (1)$$

where θ are the model parameters, $\boldsymbol{D}$ is the training corpus, $\mathrm{P}(\theta)$ is the prior of the parameters and $\mathrm{P}(\boldsymbol{D} \mid \theta)$ is the data likelihood.

In context-independent models such as Morfessor Baseline, the parameters include only the forms and probabilities of the morphs in the lexicon of the model. Morfessor Baseline and Categories-ML (CatML) (Creutz and Lagus, 2004) use a flat lexicon, in which the forms of the morphs are encoded directly as strings: each letter requires a certain number of bits to encode. Thus longer morphs are more expensive. Encoding a long morph is worthwhile only if the morph is referred to frequently enough from the words in the training data. If a certain string, let us say *segmentation*, is common enough in the training data, it is cost-effective to have it as a whole in the lexicon. Splitting it into two items, *segment* and *ation*, would double the number of pointers from the data, even if those morphs were already in the lexicon. The undersegmentation of frequent words becomes evident especially if the training data is a corpus instead of a list of unique word forms.

In contrast, Morfessor CatMAP applies a hierarchical lexicon, which makes use of the morphs that are already in the lexicon. Instead of encoding the form of *segmentation* by its 12 letters, we could just encode the form with two references to the forms of the morphs *segment* and *ation*. This may also cause errors, for example encoding *station* with *st* and *ation*.

The lexicon of Morfessor CatMAP allows but does not force hierarchical encoding for the forms: each morph has an extra parameter that indicates whether it has a hierarchical representation or not. The problem of oversegmentation, as in *st* + *ation*, is solved using the morph categories. The categories, which are states of the HMM, include stem, prefix, suffix, and a special non-morpheme category. The non-morpheme category is intended to catch segments that do not fit well into the three proper morph categories because they are fragments of a larger morph. In our example, the morph *st* cannot be a suffix as it starts the word, it is unlikely to be a prefix as it directly precedes a common suffix *ation*, and it is unlikely to be a stem as it is very short. Thus the algorithm is likely to use the non-morpheme state. The hierarchy is expanded only up to the level in which there are no non-morphemes, so the final analysis is still *station*. Without the hierarchy, the non-morphemes have to be removed heuristically, as in CatML (Creutz and Lagus, 2004).

A hierarchical lexicon presents some challenges to model training. For a standard unigram or HMM model, if you know the state and emission sequence of the training data, you can directly derive the maximum likelihood (ML) parameters of the model: a probability of a morph is proportional to the number of times it is referred to, conditional on the state in the HMM. But if the lexicon is partly hierarchical, also the references *within* the lexicon add to the reference counts, and there is no direct way to find the ML parameters even if the encoding of the training data is known. Similarly, semi-supervised learning cannot be accomplished simply by adding the counts from an annotated data set, as it is not clear when to use hierarchy instead of segmenting a word directly in the data.

Moreover, for a flat lexicon, the cost function divides into two parts that have opposing optima: the cost of the data (likelihood) is optimal when there is minimal splitting and the lexicon consists of the words in the training data, whereas the cost of the model (prior) is optimal when the lexicon is minimal and consists only of the letters. In consequence, the balance of precision and recall of the segmentation boundaries can be directly controlled by setting a weight for the data likelihood. Tuning this hyperparameter is a very simple form of supervision, but it has drastic effects on the segmentation results

(Kohonen et al., 2010). A direct control of the balance may also be useful for some applications: Virpioja et al. (2011) found that the performance of the segmentation algorithms in machine translation correlates more with the precision than the recall. The weighting approach does not work for hierarchical lexicons, for which changing the weight does not directly affect the decision whether to encode the morph with hierarchy or not.

1.2 Morfessor FlatCat

In this paper, we introduce a new member to the Morfessor family, Morfessor FlatCat. As indicated by its name, FlatCat uses a flat lexicon. Our hypothesis is that enabling semi-supervised learning is effective in compensating for the undersegmentation caused by the lack of hierarchy. In particular, semi-supervised learning can improve modeling of suffixation. In the examined languages, suffixes tend to serve syntactic purposes, such as marking case, tense, person or number. Examples are the suffix *s* marking tense and person in *she writes* and number in *stations*. Thus the suffix class is closed and has only a small number of morphemes compared to the prefix and stem categories. As a consequence, a large coverage of suffixes can be achieved already with a relatively small annotated data set.

The basic model of morphotactics in FlatCat is the same as in the CatML and CatMAP variants: a hidden Markov model with states that correspond to a word boundary and four morph categories: stem, prefix, suffix, and non-morpheme. As in CatML, we apply heuristics for removal of non-morphemes from the final segmentation. However, because FlatCat uses MAP estimation of the parameters, these heuristics are not necessary during the training for controlling the model complexity, but merely used as a post-processing step to get meaningful categories.

Modeling of morphotactics improves the segmentation of compound words, by allowing the overall level of segmentation to be increased without increasing the number of correct morphs used in incorrect positions. As the benefits of semi-supervised learning and improved morphotactics are likely to complement each other, we can expect improved performance over the semi-supervised Morfessor Baseline method. By experimental comparison to the previous Morfessor variants, we are able to shed more light on the effects of using an HMM versus unigram model for morphotactics, using a hierarchical versus flat lexicon, and exploiting small amounts of annotated training data.

2 FlatCat model and algorithms

Morfessor FlatCat uses components from the older Morfessor variants. Instead of going through all the details, we refer to the previous work and highlight only the differences. Common components between Morfessor methods are summarized in Table 1.

As a generative model, Morfessor FlatCat describes the joint distribution $\mathrm{P}(\boldsymbol{A}, \boldsymbol{W} \mid \theta)$ of words and their analyses. The words $\boldsymbol{W}$ are observed, but their analyses, $\boldsymbol{A}$, is a latent variable in the model. An analysis of a word contains its morphs and morph categories: prefix, stem, suffix, and non-morpheme.

As marginalizing over all possible analyses is generally infeasible, point estimates are used during the training. The likelihood conditioned on the current analyses is

$$\mathrm{P}(\boldsymbol{D} \mid \boldsymbol{A}, \theta) = \prod_{j=1}^{|\boldsymbol{D}|} \mathrm{P}(\boldsymbol{A}_j \mid \theta). \tag{2}$$

If m_i are the morphs in $\boldsymbol{A}_j$, c_i are the hidden states of the HMM corresponding to the categories of the morphs, and $\#$ is the word boundary, $\mathrm{P}(\boldsymbol{A}_j \mid \theta)$ is

$$\mathrm{P}(c_1 \mid \#) \prod_{i=1}^{|\boldsymbol{A}_j|} \left[\mathrm{P}(m_i \mid c_i)\, \mathrm{P}(c_{i+1} \mid c_i) \right] \mathrm{P}(\# \mid c_{|\boldsymbol{A}_j|}). \tag{3}$$

Morfessor FlatCat applies an MDL-derived prior designed to control the number of non-zero parameters. The prior is otherwise the same as in Morfessor Baseline, but it includes the usage properties from Morfessor CatMAP: the length of the morph and its right and left perplexity. The perplexity measures describe the predictability of the contexts in which the morph occurs. The emission probability of

	Morfessor method			
Component	Baseline	CatMAP	CatML	**FlatCat**
Lexicon type	Flat	Hierarchy	Flat	Flat
Morphotactics	Unigram	HMM	HMM	HMM
Estimation	MAP	MAP	ML	MAP
Semi-supervised	Implemented	Not implemented	Not implemented	Implemented

Table 1: Overview of similarities and differences between Morfessor methods.

a morph conditioned on the morph category, $\mathrm{P}(m \mid c)$, is calculated from the properties of the morphs similarly as in CatMAP.

2.1 Training algorithms

The parameters are optimized using a local search. Only a part of the parameters are optimized in each step: the parameters that are used in calculating the likelihood of a certain part, *unit*, of the corpus. Units vary in complexity, from all occurrences of a certain morph to the occurrences of a morph bigram whose context fits to certain criteria.

The algorithm tries to simultaneously find the optimal segmentation for the unit and the optimal parameters consistent with that segmentation:

$$(\boldsymbol{A}, \theta) = \underset{\textsc{Op}(\boldsymbol{A},\theta)}{\arg\min} \big\{ L(\theta, \boldsymbol{A}, \boldsymbol{D}) \big\}. \tag{4}$$

The training operators OP define the units changed by the local search and the alternative segmentations tried for each unit. There are three training operators: *split, join* and *resegment*, analogous to the similarly named stages in CatMAP.

The split operator is applied first. It targets all occurrences of a specific morph in the corpus simultaneously, attempting to split it into two parts. The whole corpus is processed by sorting the current morphs by length from shortest to longest.

The second operator attempts to join morph bigrams, grouped by the position of the bigram in the word. The position grouped bigram counts are sorted by frequency, from most to least common.

Finally, resegmenting uses the generalized Viterbi algorithm to find the currently optimal segmentation for one whole word at a time. This operator targets each corpus word in increasing order of frequency.

The heuristics used in FlatCat to remove non-morphemes from the final segmentation are the following: All consequent non-morphemes are joined together. If the resulting morph is longer than 4 characters, it is accepted as a stem. All non-morphemes preceded by a suffix and followed by only suffixes or other non-morphemes are recategorized as suffixes without joining with their neighbors. If any short non-morphemes remain, they are joined either to the preceding or following morphs (the latter only for those in the initial position).

2.2 Semi-supervised learning

Kohonen et al. (2010) found that semi-supervised learning of Morfessor models was not effective by only fixing the values of the analysis $\boldsymbol{A}$ for the annotated samples $\boldsymbol{D}_A$. Their solution was to introduce corpus likelihood weights α and β, one for the unannotated data set and one for the annotated data set. Thus, instead of optimizing the MAP estimate, Kohonen et al. (2010) minimize the cost

$$L(\theta, \boldsymbol{A}, \boldsymbol{D}, \boldsymbol{D}_A) = -\log \mathrm{P}(\theta) - \alpha \log \mathrm{P}(\boldsymbol{D} \mid \boldsymbol{A}, \theta) - \beta \log \mathrm{P}(\boldsymbol{D}_A \mid \boldsymbol{A}, \theta). \tag{5}$$

The weights can be tuned on a development set. We use the same scheme for FlatCat.

The likelihood of the annotated data is calculated using the same HMM that is used for the unannotated data. The morph properties are estimated only from the unannotated data. To ensure that the morphs required for the annotated data can be emitted, a copy of each word in the annotations is added to the

(a) English.

Method	α	β	Pre	Rec	F
U Baseline	1.0	–	.88	.59	.71
U CatMAP	–	–	.89	.51	.65
U FlatCat	1.0	–	.90	.57	.69
W Baseline	0.7	–	.83	.62	.71
W FlatCat	0.5	–	.84	.60	.70
SS Baseline	1.0	3000	.83	**.77**	.80
SS FlatCat	0.9	2000	.86	.76	.81
SS CRF+FlatCat	0.9	2000	.87	**.77**	**.82**
S CRF	–	–	**.92**	.73	.81

(b) Finnish.

Method	α	β	Pre	Rec	F
U Baseline	1.0	–	.84	.38	.53
U CatMAP	–	–	.76	.51	.61
U FlatCat	1.0	–	.84	.38	.52
W Baseline	.02	–	.62	.54	.58
W FlatCat	.015	–	.66	.52	.58
SS Baseline	.1	15000	.75	.72	.73
SS FlatCat	.2	1500	.79	.71	.75
SS CRF+FlatCat	.2	2500	.82	**.76**	.79
S CRF	–	–	**.88**	.74	**.80**

Table 2: Boundary Precision and Recall results in comparison to gold standard segmentation. Abbreviations have been used for Unsupervised (U), likelihood weighted (W), semi-supervised (SS) and fully supervised (S) methods. Best results for each measure have been hilighted using boldface.

unannotated data. This unannotated copy is loosely linked to the annotated word: operations that would result in the removal of a morph required for the annotations from the lexicon cannot be selected, as such an operation would have infinite cost.

3 Experiments

We compare Morfessor FlatCat[1] to two previous Morfessor methods and a fully supervised discriminative segmentation method. The Morfessor methods used as references are the CatMAP[2] and Baseline[3] implementations by Creutz and Lagus (2005) and Virpioja et al. (2013), respectively. Virpioja et al. (2013) implements the semi-supervised method described by Kohonen et al. (2010). For a supervised discriminative model, we use a character-level conditional random field (CRF) implementation by Ruokolainen et al. (2013)[4].

We use the English, Finnish and Turkish data sets from Morpho Challenge 2010 (Kurimo et al., 2010b). They include large unannotated word lists, one thousand annotated words for training, 700–800 annotated words for parameter tuning, and 10×1000 annotated words for testing.

For evalution, we use the BPR score by Virpioja et al. (2011). The score calculates the precision (Pre), recall (Rec), and F_1-score (F) of the predicted morph boundaries compared to a linguistic gold standard. In the presence of alternative gold standard analyses, we weight each alternative equally.

We also report the mean average precision from the English and Finnish information retrieval (IR) tasks of the Morpho Challenge. The Lemur Toolkit (Ogilvie and Callan, 2001) with Okapi BM25 ranking was used. The Finnish data consists of 55K documents, 50 test queries and 23K binary relevance assessments. The English data consists of 170K documents, 50 test queries and 20K binary relevance assessments. The domain of both data sets is short newspaper articles. All word forms in both the corpora and the queries were replaced by the morphological segmentation to be evaluated.

Morfessor FlatCat is a pipeline method that refines an initial segmentation given as input. We try two different initializations for the semi-supervised setting: initializing with the segmentation produced by semi-supervised Morfessor Baseline, and initializing with the CRF segmentation. All unsupervised and likelihood-weighted results are initialized with the corresponding Baseline output.

All methods were trained using word types. The weight and perplexity threshold parameters were optimized separately for each method, using a grid search with the held-out data set. The supervised CRF method was trained using the one thousand word annotated training data set.

[1]Available at `https://github.com/aalto-speech/flatcat`
[2]Available at `http://www.cis.hut.fi/projects/morpho/morfessorcatmap.shtml`
[3]Available at `https://github.com/aalto-speech/morfessor`
[4]Available at `http://users.ics.aalto.fi/tpruokol/`

Method	α	β	Pre	Rec	F
U Baseline	1.0	–	.85	.36	.51
U CatMAP	–	–	.83	.50	.62
U FlatCat	1.0	–	.87	.36	.51
W Baseline	0.1	–	.71	.41	.52
W FlatCat	0.3	–	.88	.38	.53
SS Baseline	0.4	2000	.86	.60	.71
SS FlatCat	0.8	2666	.87	.59	.70
SS CRF+FlatCat	1.0	3000	.87	**.61**	**.72**
S CRF	–	–	**.89**	.58	.70

Table 3: Boundary Precision and Recall results in comparison to gold standard segmentation for Turkish. Abbreviations have been used for Unsupervised (U), likelihood weighted (W), semi-supervised (SS) and fully supervised (S) methods. Best results for each measure have been hilighted using boldface.

3.1 Comparison to linguistic gold standards

The results of the BPR evaluations are shown in Tables 2 (English, Finnish) and 3 (Turkish). Semi-supervised FlatCat initialized using CRF achieves the highest F-score for both the English and Turkish data sets. The difference between the highest and second-highest scoring methods is statistically significant for Finnish and Turkish, but not for English (Wilcoxon signed-rank test, $p < 0.01$).

Table 4 shows BPR for subsets of words consisting of different morph category patterns. Each subset consists of 500 words from the English or Finnish gold standard, with one of five selected morph patterns as the only valid analysis. The subsets consist of words with the following morph patterns: words that should not be segmented (STM), compound words consisting of exactly two stems (STM + STM), a prefix followed by a stem (PRE + STM), a stem followed by a single suffix (STM + SUF) and a stem and exactly two suffixes (STM + SUF + SUF). For the STM pattern only precision is reported, as recall is not defined for an empty set of true boundaries.

The fact that semi-supervised FlatCat compares well against CatMAP in recall, for all morph patterns and for the test set as a whole, indicates that supervision indeed is effective in compensating for the undersegmentation caused by the lack of hierarchy in the lexicon. The benefit of modeling morphotactics can be seen in improved precision for the STM + STM (for English and Finnish) and PRE + STM (for Finnish) patterns when comparing against semi-supervised Baseline. The more aggressive segmentation of Baseline gives better results for the English PRE + STM subset than for Finnish due to the shortness of the English prefixes (on average 3.6 letters for the English and 5.3 for the Finnish subset). While not directly observable in Table 4, a large part of the improvement over semi-supervised Baseline is explained by that FlatCat does not use suffix-like morphs in incorrect positions.

Initializing the FlatCat model with CRF segmentation improves the F-scores in all subsets compared to the initialization with Morfessor Baseline. While FlatCat cannot keep the accuracy of the suffix boundaries at as high level as CRF, it clearly improves the stem splitting.

3.2 Information retrieval

Stemming has been shown to improve IR results (Kurimo et al., 2009), by removing inflection that is often not relevant to the query. The morph categories make it possible to simulate stemming by removing morphs categorized as prefixes or suffixes. As longer affixes are more likely to be meaningful, we limited the affix removal to morphs of at most 3 letters. For methods that use morph categories, we report two IR results: the first using all the data and the second with short affix removal (SAR) applied.

In the IR results, we include the topline methods from Morpho Challenge: Snowball Porter stemmer (Porter, 1980) for English and "TWOL first" for Finnish. The latter selects the lemma from the first of the possible analyses given by the morphological analyzer FINTWOL (Lingsoft, Inc.) based on the

(a) English.

	STM	STM + STM			PRE + STM			STM + SUF			STM + SUF + SUF		
Method	Pre	Pre	Rec	F	Pre	Rec	F	Pre	Rec	F	Pre	Rec	F
U CatMAP	**.90**	.94	.63	.75	**.91**	.64	.75	.87	.45	.59	.90	.51	.65
SS Baseline	.64	.93	.77	.84	.82	**.74**	**.77**	.83	.86	.84	.91	.79	.85
SS FlatCat	.68	.94	.65	.77	.78	.62	.69	.86	.88	.87	.94	.79	.86
SS CRF+FlatCat	.68	**.95**	**.78**	**.86**	.78	.66	.72	.87	.89	.88	.94	.80	.87
S CRF	.78	.94	.72	.81	.85	.59	.69	**.92**	**.91**	**.91**	**.95**	**.82**	**.88**

(b) Finnish.

	STM	STM + STM			PRE + STM			STM + SUF			STM + SUF + SUF		
Method	Pre	Pre	Rec	F	Pre	Rec	F	Pre	Rec	F	Pre	Rec	F
U CatMAP	**.77**	.90	**.97**	**.94**	.88	**.96**	**.92**	.67	.46	.54	.68	.38	.49
SS Baseline	.50	.82	.88	.85	.73	.83	.78	.64	.85	.73	.76	.78	.77
SS FlatCat	.49	**.91**	.95	.93	.80	.89	.85	.67	.84	.75	.77	.75	.76
SS CRF+FlatCat	.53	**.91**	.96	**.94**	.84	.94	.88	.71	.88	.79	.80	.79	.79
S CRF	.68	.88	.91	.89	**.90**	.91	.91	**.83**	**.91**	**.87**	**.91**	**.85**	**.88**

Table 4: Results of BPR experiments with different morph category patterns. Best results for each measure have been hilighted using boldface.

two-level model by Koskenniemi (1983). As baseline results we also include unsegmented word forms and truncating each word after the first five letters (First 5).

The results of the IR experiment are shown in Table 5. FlatCat provides the highest score for Finnish. The English scores are similar to those of the semi-supervised Baseline. FlatCat performs better than CRF for both languages. This is explained by the higher level of consistency in the segmentations produced by FlatCat, which makes the resulting morphs more useful as query terms. The number of morphs in the lexicons of FlatCat initialized using CRF are 108 391 (English), 46 123 (Finnish) and 74 193 (Turkish), which is much smaller than the respective morph lexicon sizes counted from the CRF segmentation: 339 682 (English), 396 869 (Finnish) and 182 356 (Turkish). This decrease in lexicon size indicates a more structured segmentation.

The IR performance of semi-supervised FlatCat benefits from the removal of short affixes for English when initialized by CRF, and Finnish for both initializations. It also improves the results of unsupervised FlatCat and CatMAP for Finnish, but lowers the precision for English. A possible explanation is that the unsupervised methods do not analyze the suffixes with a high enough accuracy.

4 Conclusions

We have introduced a new variant of the Morfessor method, Morfessor FlatCat. It predicts both morphs and their categories based on unannotated data, but also annotated training data can be provided. It was shown to outperform earlier Morfessor methods in the semi-supervised learning task for English, Finnish and Turkish.

The purely supervised CRF-based segmentation method proposed by Ruokolainen et al. (2013) outperforms FlatCat for Finnish and reaches the same level for English. However, we show that a discriminative model such as CRF gives inconsistent segmentations that do not work as well in a practical application: In English and Finnish information retrieval tasks, FlatCat clearly outperformed the CRF-based segmentation.

We see two major directions for future work. Currently Morfessor FlatCat, like most Morfessor methods, assumes that words in a sentence occur independently. Making use of the sentence context in which words occur would, however, allow making Part-Of-Speech -like distinctions. These distinctions could

(a) English.

Rank		Method	SAR	MAP
1	–	Snowball Porter	–	0.4092
2	SS	Baseline	–	0.3855
3	**SS**	**FlatCat**	**No**	**0.3837**
4	**SS**	**FlatCat**	**Yes**	**0.3821**
5	**SS**	**CRF+FlatCat**	**Yes**	**0.3810**
6	**SS**	**CRF+FlatCat**	**No**	**0.3788**
7	S	CRF	–	0.3771
8	W	Baseline	–	0.3761
9	U	Baseline	–	0.3695
10	U	CatMAP	No	0.3682
11	U	CatMAP	Yes	0.3653
12	**W**	**FlatCat**	**No**	**0.3651**
13	–	(First 5)	–	0.3648
14	**W**	**FlatCat**	**Yes**	**0.3606**
15	**U**	**FlatCat**	**No**	**0.3486**
16	**U**	**FlatCat**	**Yes**	**0.3451**
17	–	(Words)	–	0.3303

(b) Finnish.

Rank		Method	SAR	MAP
1	**W**	**FlatCat**	**No**	**0.5057**
2	**W**	**FlatCat**	**Yes**	**0.5029**
3	**SS**	**FlatCat**	**Yes**	**0.4987**
4	–	TWOL first	–	0.4973
5	**SS**	**CRF+FlatCat**	**Yes**	**0.4912**
6	U	CatMAP	Yes	0.4884
7	U	CatMAP	No	0.4865
8	**SS**	**CRF+FlatCat**	**No**	**0.4826**
9	**SS**	**FlatCat**	**No**	**0.4821**
10	–	(First 5)	–	0.4757
11	SS	Baseline	–	0.4722
12	S	CRF	–	0.4660
13	W	Baseline	–	0.4582
14	U	Baseline	–	0.4378
15	**U**	**FlatCat**	**Yes**	**0.4349**
16	**U**	**FlatCat**	**No**	**0.4334**
17	–	(Words)	–	0.3483

Table 5: Information Retrieval results. Results of the method presented in this paper are hilighted using boldface. Mean Average Precision is abbreviated as MAP. Short affix removal is abbreviated as SAR.

help disambiguate inflections of different lexemes that have the same surface form but should be analyzed differently (Can and Manandhar, 2013).

The second direction is removal of the assumption that a morphology consists only of concatenative processes. Introducing transformations to model allomorphy in a similar manner as Kohonen et al. (2009) would allow finding the shared abstract morphemes underlying different allomorphs. This could be especially beneficial in information retrieval and machine translation applications.

Acknowledgments

This research has been supported by European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreement n°287678 and the Academy of Finland under the Finnish Centre of Excellence Program 2012–2017 (grant n°251170) and the LASTU Programme (grants n°256887 and 259934). The experiments were performed using computer resources within the Aalto University School of Science "Science-IT" project. We thank Teemu Ruokolainen for his help with the experiments.

References

Burcu Can and Suresh Manandhar. 2013. Dirichlet processes for joint learning of morphology and PoS tags. In *Proceedings of the International Joint Conference on Natural Language Processing*, pages 1087–1091, Nagoya, Japan, October.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In Mike Maxwell, editor, *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 21–30, Philadelphia, PA, USA, July. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2004. Induction of a simple morphology for highly-inflecting languages. In *Proceedings of the Seventh Meeting of the ACL Special Interest Group in Computational Phonology*, pages 43–51, Barcelona, Spain, July. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In Timo Honkela, Ville Könönen, Matti Pöllä, and Olli Simula, editors, *Proceedings of AKRR'05,*

International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning, pages 106–113, Espoo, Finland, June. Helsinki University of Technology, Laboratory of Computer and Information Science.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1):3:1–3:34, January.

Harald Hammarström and Lars Borin. 2011. Unsupervised learning of morphology. *Computational Linguistics*, 37(2):309–350, June.

Oskar Kohonen, Sami Virpioja, and Mikaela Klami. 2009. Allomorfessor: Towards unsupervised morpheme analysis. In *Evaluating Systems for Multilingual and Multimodal Information Access: 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Aarhus, Denmark, September 17–19, 2008, Revised Selected Papers*, volume 5706 of *Lecture Notes in Computer Science*, pages 975–982. Springer Berlin / Heidelberg, September.

Oskar Kohonen, Sami Virpioja, and Krista Lagus. 2010. Semi-supervised learning of concatenative morphology. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 78–86, Uppsala, Sweden, July. Association for Computational Linguistics.

Kimmo Koskenniemi. 1983. *Two-level morphology: A general computational model for word-form recognition and production*. Ph.D. thesis, University of Helsinki.

Mikko Kurimo, Sami Virpioja, Ville T. Turunen, Graeme W. Blackwood, and William Byrne. 2009. Overview and results of Morpho Challenge 2009. In *Working Notes for the CLEF 2009 Workshop*, Corfu, Greece, September.

Mikko Kurimo, Sami Virpioja, Ville Turunen, and Krista Lagus. 2010a. Morpho Challenge 2005-2010: Evaluations and results. In Jeffrey Heinz, Lynne Cahill, and Richard Wicentowski, editors, *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 87–95, Uppsala, Sweden, July. Association for Computational Linguistics.

Mikko Kurimo, Sami Virpioja, and Ville T. Turunen. 2010b. Overview and results of Morpho Challenge 2010. In *Proceedings of the Morpho Challenge 2010 Workshop*, pages 7–24, Espoo, Finland, September. Aalto University School of Science and Technology, Department of Information and Computer Science. Technical Report TKK-ICS-R37.

Paul Ogilvie and James P Callan. 2001. Experiments using the Lemur toolkit. In *TREC*, volume 10, pages 103–108.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14:465–471.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 29–37, Sofia, Bulgaria, August. Association for Computational Linguistics.

Sami Virpioja, Ville T Turunen, Sebastian Spiegler, Oskar Kohonen, and Mikko Kurimo. 2011. Empirical comparison of evaluation methods for unsupervised learning of morphology. *Traitement Automatique des Langues*, 52(2):45–90.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. 2013. Morfessor 2.0: Python implementation and extensions for Morfessor Baseline. Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, Department of Signal Processing and Acoustics, Aalto University.

Publication II

Teemu Ruokolainen, Oskar Kohonen, Kairit Sirts, Stig-Arne Grönroos, Mikko Kurimo, and Sami Virpioja. A comparative study on minimally supervised morphological segmentation. *Computational Linguistics*, Vol. 42, No. 1, pages 91–120, Mar 2016.

A Comparative Study of Minimally Supervised Morphological Segmentation

Teemu Ruokolainen*
Aalto University

Oskar Kohonen**
Aalto University

Kairit Sirts†
Tallinn University of Technology

Stig-Arne Grönroos*
Aalto University

Mikko Kurimo*
Aalto University

Sami Virpioja**
Aalto University

This article presents a comparative study of a subfield of morphology learning referred to as **minimally supervised morphological segmentation**. *In morphological segmentation, word forms are segmented into morphs, the surface forms of morphemes. In the minimally supervised data-driven learning setting, segmentation models are learned from a small number of manually annotated word forms and a large set of unannotated word forms. In addition to providing a literature survey on published methods, we present an in-depth empirical comparison on three diverse model families, including a detailed error analysis. Based on the literature survey, we conclude that the existing methodology contains substantial work on generative morph lexicon-based approaches and methods based on discriminative boundary detection. As for which approach has been more successful, both the previous work and the empirical evaluation presented here strongly imply that the current state of the art is yielded by the discriminative boundary detection methodology.*

* Department of Signal Processing and Acoustics, Otakaari 5 A, FI-02150 Espoo, Finland. E-mail: {teemu.ruokolainen, stig-arne.gronroos, mikko.kurimo}@aalto.fi.
** Department of Information and Computer Science, Konemiehentie 2, FI-02150 Espoo, Finland. E-mail: {oskar.kohonen, sami.virpioja}@aalto.fi.
† Institute of Cybernetics, Akadeemia tee 21 EE-12618 Tallin, Estonia. E-mail: sirts@phon.ioc.ee.

Submission received: 15 September 2014; revised version received: 7 October 2015; accepted for publication: 15 November 2015

doi:10.1162/COLI_a_00243

1. Introduction

This article discusses a subfield of morphology learning referred to as **morphological segmentation**, in which word forms are segmented into **morphs**, the surface forms of **morphemes**. For example, consider the English word *houses* with a corresponding segmentation *house+s*, where the segment *house* corresponds to the word stem and the suffix *-s* marks the plural number. Although this is a major simplification of the diverse morphological phenomena present in languages, this type of analysis has nevertheless been of substantial interest to computational linguistics, beginning with the pioneering work on morphological learning by Harris (1955). As for automatic language processing, such segmentations have been found useful in a wide range of applications, including speech recognition (Hirsimäki et al. 2006; Narasimhan et al. 2014), information retrieval (Turunen and Kurimo 2011), machine translation (de Gispert et al. 2009; Green and DeNero 2012), and word representation learning (Luong, Socher, and Manning 2013; Qiu et al. 2014).

Since the early work of Harris (1955), most research on morphological segmentation has focused on **unsupervised** learning, which aims to learn the segmentation from a list of unannotated (unlabeled) word forms. The unsupervised methods are appealing as they can be applied to any language for which there exists a sufficiently large set of unannotated words in electronic form. Consequently, such methods provide an inexpensive means of acquiring a type of morphological analysis for low-resource languages as motivated, for example, by Creutz and Lagus (2002). The unsupervised approach and learning setting has received further popularity because of its close relationship with the unsupervised **word segmentation** problem, which has been viewed as a realistic setting for theoretical study of language acquisition (Brent 1999; Goldwater 2006).

Although development of novel unsupervised model formulations has remained a topic of active research (Poon, Cherry, and Toutanova 2009; Monson, Hollingshead, and Roark 2010; Spiegler and Flach 2010; Lee, Haghighi, and Barzilay 2011; Sirts and Goldwater 2013), recent work has also shown a growing interest towards **semi-supervised** learning (Poon, Cherry, and Toutanova 2009; Kohonen, Virpioja, and Lagus 2010; Sirts and Goldwater 2013; Grönroos et al. 2014; Ruokolainen et al. 2014). In general, the aim of semi-supervised learning is to acquire high-performing models utilizing both unannotated as well as annotated data (Zhu and Goldberg 2009). In morphological segmentation, the annotated data sets are commonly small, on the order of a few hundred word forms. We refer to this learning setting with such a small amount of supervision as **minimally supervised** learning. In consequence, similar to the unsupervised methods, the minimally supervised techniques can be seen as a means of acquiring a type of morphological analysis for under-resourced languages.

Individual articles describing novel methods typically contain a comparative discussion and empirical evaluation between one or two preceding approaches. Therefore, what is currently lacking from the literature is a summarizing comparative study on the published methodology as a whole. Moreover, the literature currently lacks discussion on error analysis. A study on the error patterns produced by varying approaches could inform us about their potential utility in different tasks. For example, if an application requires high-accuracy compound splitting, one could choose to apply a model with a good compound-splitting capability even if its affix accuracy does not reach state of the art. The purpose of this work is to address these issues.

Our main contributions are as follows. First, we present a literature survey on morphological segmentation methods applicable in the minimally supervised learning setting. The considered methods include unsupervised techniques that learn solely from

unannotated data, supervised methods that utilize solely annotated data, and semi-supervised approaches that utilize both unannotated and annotated data. Second, we perform an extensive empirical evaluation of three diverse method families, including a detailed error analysis. The approaches considered in this comparison are variants of the Morfessor algorithm (Creutz and Lagus 2002, 2005, 2007; Kohonen, Virpioja, and Lagus 2010; Grönroos et al. 2014), the adaptor grammar framework (Sirts and Goldwater 2013), and the conditional random field method (Ruokolainen et al. 2013, 2014). We hope the presented discussion and empirical evaluation will be of help for future research on the considered task.

The rest of the article is organized as follows. In Section 2, we provide an overview of related studies. We then provide a literature survey of published morphological segmentation methodology in Section 3. Experimental work is presented in Section 4. Finally, we provide a discussion on potential directions for future work and conclusions on the current work in Sections 5 and 6, respectively.

2. Related Work

Hammarström and Borin (2011) presented a literature survey on unsupervised learning of morphology, including methods for learning morphological segmentation. Whereas the discussion provided by Hammarström and Borin focuses mainly on linguistic aspects of morphology learning, our work is strongly rooted in machine learning methodology and empirical evaluation. In addition, whereas Hammarström and Borin focus entirely on unsupervised learning, our work considers a broader range of learning paradigms. Therefore, although related, Hammarström and Borin and our current presentation are complementary in that they have different focus areas.

In addition to the work of Hammarström and Borin (2011), we note that there exists some established forums on morphology learning. First, we mention the ACL Special Interest Group on Computational Morphology and Phonology (SIGMORPHON), which has regularly organized workshops on the subject since 2002. As for specifically morphology learning, we refer to the Morpho Challenge competitions organized since 2005 at Aalto University (formerly known as Helsinki University of Technology). Although these events have been successful in providing a publication and discussion venue for researchers interested in the topic, they have not given birth to comparative studies or survey literature. For example, whereas the publications on the Morpho Challenge (Kurimo et al. 2009; Kurimo, Virpioja, and Turunen 2010) discuss the competition results, they nevertheless do not attempt to provide any insight on the fundamental differences and similarities of the participating methods.

3. Methods

This section provides a detailed review of our methodology. We begin by describing varying morphological representations, including segmentation, and the minimally supervised learning setting in Sections 3.1 and 3.2, respectively. We then provide a literature survey and comparative discussion on a range of methods in Section 3.3.

3.1 On Learning Morphological Representations

In what follows, we briefly characterize morphological segmentation with respect to alternative morphological representations, particularly the **full morphological analysis**. To this end, consider the exemplar segmentations and full analyses for Finnish

Table 1
Morphological segmentation versus full morphological analysis for exemplar Finnish word forms. The full analysis consists of word lemma (basic form), part-of-speech, and fine-grained labels.

word form	full analysis	segmentation
auto (car)	auto+N+Sg+Nom	auto
autossa (in car)	auto+N+Sg+Ine	auto+ssa
autoilta (from cars)	auto+N+Pl+Abl	auto+i+lta
autoilta (car evening)	auto+N+Sg+Nom+# ilta+N+Sg+Nom	auto+ilta
maantie (highway)	maantie+N+Sg+Nom	maantie
	maa+N+Sg+Gen+# tie+N+Sg+Nom	maa+n+tie
sähköauto (electric car)	sähköauto+N+Sg+Nom	sähköauto
	sähkö+N+Sg+Nom+# auto+N+Sg+Nom	sähkö+auto

word forms in Table 1, where the full analyses are provided by the rule-based OMorFi analyzer developed by Pirinen (2008). Note that it is typical for word forms to have alternative analyses and/or meanings that cannot be disambiguated without sentential context. Evidently, the level of detail in the full analysis is substantially higher compared with the segmentation, as it contains lemmatization as well as morphological tagging, whereas the segmentation consists of only segment boundary positions. Consequently, because of this simplification, morphological segmentation has been amenable to unsupervised machine learning methodology, beginning with the work of Harris (1955). Meanwhile, the majority of work on learning of full morphological analysis has used supervised methodology (Chrupala, Dinu, and van Genabith 2008). Lastly, there have been numerous studies on statistical learning of intermediate forms of segmentation and full analysis (Lignos 2010; Virpioja, Kohonen, and Lagus 2010) as well as alternative morphological representations (Yarowsky and Wicentowski 2000; Schone and Jurafsky 2001; Neuvel and Fulop 2002; Johnson and Martin 2003).

As for language processing, learning segmentation can be advantageous compared with learning full analyses. In particular, learning full analysis in a supervised manner typically requires up to tens of thousands of manually annotated sentences. A low-cost alternative, therefore, could be to learn morphological segmentation from unannotated word lists and a handful of annotated examples. Importantly, segmentation analysis has been found useful in a range of applications, such as speech recognition (Hirsimäki et al. 2006; Narasimhan et al. 2014), information retrieval (Turunen and Kurimo 2011), machine translation (de Gispert et al. 2009; Green and DeNero 2012), and word representation learning (Luong, Socher, and Manning 2013; Qiu et al. 2014).

Despite its intuitiveness, it should be noted that the segmentation representation is not equally applicable to all languages. To this end, consider the terms **isolative** and **synthetic** languages. In languages with a high amount of isolating morphological properties, word forms tend to comprise their own morphemes. Meanwhile, in heavily synthetic languages, words tend to contain multiple morphemes. Synthetic languages can be described further according to their **agglutinative** (**concatenative**) and **fusional** properties. In the former, the morphs tend to have clear boundaries between them whereas in the latter, the morphs tend to be indistinguishable. For examples of agglutinative and fusional word formation, consider the English verbs *played* (past tense of *play*) and *sang* (past tense of *sing*). Where the previous can be effortlessly

divided into two segments as *play+ed* (STEM + PAST TENSE), there are no such distinct boundaries in the latter. Generally, languages with synthetic properties mix concatenative and fusional schemes and contain agglutinative properties to varying degrees. Morphological segmentation can be most naturally applied to highly agglutinative languages.

Morphologically ambiguous word forms are common especially in highly synthetic languages. Even without disambiguation based on sentential context, providing all correct alternatives could be useful for some downstream applications, such as information retrieval. Statistical methods can usually provide n-best segmentations; for example, Morfessor (Creutz and Lagus 2007) and CRFs (Ruokolainen et al. 2013) by using n-best Viterbi algorithm and adaptor grammar (Sirts and Goldwater 2013) by collecting the variations in the posterior distribution samples. Although there is no evident way to decide the correct number of alternatives for a particular word form, n-best lists might be useful whenever recall (including the correct answers) is more important than precision (excluding any incorrect answers). The Morpho Challenge competitions have allowed providing alternative segmentations for the submitted methods, but no clear developments have been reported. In fact, even in the reference results based on the gold standard segmentations, selecting all alternative segmentations has performed slightly worse in the information retrieval tasks than taking only the first segmentation (Kurimo, Virpioja, and Turunen 2010).

3.2 Minimally Supervised Learning Settings

In data-driven morphological segmentation, our aim is to learn segmentation models from training data. Subsequent to training, the models provide segmentations for given word forms. In the minimally supervised learning setting, as defined here, the models are estimated from annotated and unannotated word forms. We denote the annotated data set comprising word forms with their corresponding segmentation as $\mathcal{D}$ and the unannotated data set comprising raw word forms as $\mathcal{U}$. Typically, the raw word forms can be obtained easily and, consequently, $\mathcal{U}$ can contain millions of word forms. Meanwhile, acquiring the annotated data $\mathcal{D}$ requires manual labor and, therefore, typically contains merely hundreds or thousands of word forms. For an illustration of $\mathcal{D}$ and $\mathcal{U}$, see Table 2.

Table 2
Examples of annotated and unannotated data, $\mathcal{D}$ and $\mathcal{U}$, respectively. Typically, $\mathcal{U}$ can contain hundreds of thousands or millions of word forms, whereas $\mathcal{D}$ contains merely hundreds or thousands of word forms.

$\mathcal{D}$	$\mathcal{U}$
anarch + ist + s	actions
bound + ed	bilinguals
conting + ency	community
de + fame	disorders
entitle + ment	equipped
fresh + man	faster
...	...

We consider three machine learning approaches applicable in the minimally supervised learning setting, namely, unsupervised, supervised, and semi-supervised learning. In unsupervised learning, the segmentation models are trained on solely unannotated data $\mathcal{U}$. Meanwhile, supervised models are trained from solely the annotated data $\mathcal{D}$. Finally, the aim of semi-supervised learning is to utilize both the available unannotated and annotated data. Because the semi-supervised approach utilizes the largest amount of data, it is expected to be most suitable for acquiring high segmentation accuracy in the minimally supervised learning setting.

Lastly, we note that the unsupervised learning framework can be understood in a strict or non-strict sense, depending on whether the applied methods are allowed to use annotated data $\mathcal{D}$ for hyperparameter tuning. Although the term unsupervised learning itself suggests that such adjusting is infeasible, this type of tuning is nevertheless common (Creutz et al. 2007; Çöltekin 2010; Spiegler and Flach 2010; Sirts and Goldwater 2013). In addition, the minimally supervised learning setting explicitly assumes a small amount of available annotated word forms. Consequently, in the remainder of this article, all discussion on unsupervised methods refers to unsupervised learning in the non-strict sense.

3.3 Algorithms

Here we provide a literature survey on proposed morphological segmentation methods applicable in the minimally supervised learning setting. We place particular emphasis on three method families, namely, the Morfessor algorithm (Creutz and Lagus 2002, 2005, 2007; Kohonen, Virpioja, and Lagus 2010; Grönroos et al. 2014), the adaptor grammar framework (Sirts and Goldwater 2013), and conditional random fields (Ruokolainen et al. 2013, 2014). These approaches are the subject of the empirical evaluation presented in Section 4. We present individual method descriptions in Section 3.3.1. Subsequently, Section 3.3.2 provides a summarizing discussion, the purpose of which is to gain insight on the fundamental differences and similarities between the varying approaches.

3.3.1 Descriptions

Morfessor. We begin by describing the original, unsupervised Morfessor method family (Creutz and Lagus 2002, 2005, 2007). We then discuss the later, semi-supervised extensions (Kohonen, Virpioja, and Lagus 2010; Grönroos et al. 2014). In particular, we review the extension of Morfessor Baseline to semi-supervised learning by using a weighted generative model (Kohonen, Virpioja, and Lagus 2010), and then discuss the most recent Morfessor variant, FlatCat (Grönroos et al. 2014). Finally, we discuss some general results from the literature on semi-supervised learning with generative models.

The unsupervised Morfessor methods are based on a generative probabilistic model that generates the observed word forms $x_i \in \mathcal{U}$ by concatenating morphs $x_i = m_{i1} \circ m_{i2} \circ \cdots \circ m_{in}$. The morphs are stored in a **morph lexicon**, which defines the probability of each morph $P(m \mid \theta)$ given some parameters θ. The Morfessor learning problem is to find a morph lexicon that strikes an optimal balance between encoding the observed word forms concisely and, at the same time, having a concise morph lexicon. To this end, Morfessor utilizes a prior distribution $P(\theta)$ over morph lexicons, derived from the Minimum Description Length principle (Rissanen 1989), that favors lexicons that contain fewer, shorter morphs. This leads to the following minimization problem

that seeks to balance the conciseness of the lexicon with the conciseness of the observed corpus encoded with the lexicon:

$$\theta^* = \arg\min_{\theta} L(\theta, \mathcal{U}) = \arg\min_{\theta} \{-\ln P(\theta) - \ln P(\mathcal{U} \mid \theta)\}, \quad (1)$$

The optimization problem in Equation (1) is complicated by the fact that each word in the corpus $\mathcal{U}$ can be generated by different combinations of morphs, defined by the set of segmentations of that word. This introduces a nuisance parameter z for the segmentation of each word form, where $P(\mathcal{U} \mid \theta) = \sum_z P(\mathcal{U} \mid z, \theta) P(z)$. Because of this summation, the expression cannot be solved analytically, and iterative optimization must be used instead.

The unsupervised Morfessor variants differ in the following ways: first, whether all morphs belong to a single category or the categories PREFIX, STEM, and SUFFIX are used; secondly, if the model utilizes a lexicon that is **flat** or **hierarchical**. In a flat lexicon, morphs can only be encoded by combining letters, whereas in a hierarchical lexicon pre-existing morphs can be used for storing longer morphs. Thirdly, the parameter estimation and inference methods differ. Parameters are estimated using greedy local search or iterative batch procedures while inference is performed with either Viterbi decoding or heuristic procedures.

The earliest Morfessor method, referred to as Morfessor Baseline, has been extended to semi-supervised learning by Kohonen, Virpioja, and Lagus (2010). In contrast, the later methods, namely, Categories-ML and Categories-MAP, have not been extended, as they use either hierarchical lexicons or training procedures that make them less amenable to semi-supervised learning. However, recently Grönroos et al. (2014) proposed a new Morfessor variant that uses morph categories in combination with a flat lexicon, and can therefore apply the semi-supervised learning technique of Kohonen, Virpioja, and Lagus.

We begin the description of the semi-supervised extension to Morfessor Baseline (Creutz and Lagus 2002, 2007) by reviewing its generative model. Morfessor Baseline utilizes a model in which word forms are generated by concatenating morphs, all of which belong to the same category. It utilizes a flat morph lexicon $P(m \mid \theta)$ that is simply a multinomial distribution over morphs m, according to the probabilities given by the parameter vector θ. The utilized prior penalizes storing long morphs in the lexicon by assigning each stored morph a cost that depends most strongly on the morph length in letters. A morph is considered to be stored if the lexicon assigns it a nonzero probability. The parameter estimation for θ finds a local optimum utilizing greedy local search. The search procedure approximates the optimization problem in Equation (1) by assuming that, for each word form x_i, its corresponding segmentation distribution $P(z_i)$ has all its mass concentrated to a single segmentation z_i. The parameter estimation is then performed by locally searching each word for the segmentation that yields the best value of the cost function in Equation (1). The process is repeated for all words in random order until convergence. Subsequent to learning, the method predicts the segmentation of a word form by selecting the segmentation with the most probable sequence of morphs using an extension of the Viterbi algorithm.

Semi-supervised learning is in principle trivial for a generative model: For the labeled word forms $\mathcal{D}$, the segmentation is fixed to its correct value, and for the unlabeled forms $\mathcal{U}$ the standard parameter estimation procedure is applied. However, Kohonen,

Virpioja, and Lagus (2010) failed to achieve notable improvements in this fashion, and consequently replaced the minimized function L in Equation (1) with

$$L(\theta, z, \mathcal{U}, \mathcal{D}) = -\ln P(\theta) - \alpha \times \ln P(\mathcal{U} \mid \theta) - \beta \times \ln P(\mathcal{D} \mid \theta). \quad (2)$$

Such weighted objectives were used earlier in combination with generative models by, for example, Nigam et al. (2000). The semi-supervised training procedure then adjusts the weight values α and β. The absolute values of the weights control the cost of encoding a morph in the training data with respect to the cost of adding a new morph to the lexicon, and their ratio controls how much weight is placed on the annotated data with respect to the unannotated data. When the hyperparameters α and β are fixed, the lexicon parameters θ can be optimized with the same greedy local search procedure as in the unsupervised Morfessor Baseline. The weights can then be optimized with a grid search and by choosing the model with the best evaluation score on a held-out development set. Although this modification is difficult to justify from the perspective of generative modeling, Kohonen, Virpioja, and Lagus show that in practice it can yield performance improvements. From a theoretical point of view, it can be seen as incorporating discriminative training techniques when working with a generative model by optimizing for segmentation performance rather than maximum a posteriori probability. However, only the hyperparameters are optimized in this fashion, whereas the lexicon parameters are still learned within the generative model framework.

The semi-supervised learning strategy described here is simple to apply if the objective function in Equation (1) can be factored to parts that encode the morphs using letters and encode the training corpus using the morphs. For some models of the Morfessor family this is not possible because of the use of a hierarchical lexicon, where morphs can be generated from other morphs as well as from individual letters. In particular, this includes the well-performing Categories-MAP variant (Creutz et al. 2007). In contrast to Morfessor Baseline, the Categories-MAP and the preceding Categories-ML method use a hidden Markov model to produce the observed words, where the states are given by STEM, PREFIX, SUFFIX categories as well as an internal non-morpheme category. A recent development is Morfessor FlatCat by Grönroos et al. (2014), which uses the hidden Markov model structure and morph categories in combination with a flat lexicon, thus allowing semi-supervised learning in the same fashion as for Morfessor Baseline.

In general, the key idea behind using the weighted objective function in Equation (2) for semi-supervised learning is that the hyperparameters α and β can be used to explicitly control the influence of the unannotated data on the learning. Similar semi-supervised learning strategies have also been used in other problems. For classification with generative models, it is known that adding unlabeled data to a model trained with labeled data can degrade performance (Cozman et al. 2003; Cozman and Cohen 2006). In particular, this can be the case if the generative model does not match the generating process, something that is difficult to ensure in practice. Recently, this phenomenon was analyzed in more detail by Fox-Roberts and Rosten (2014), who show that, although the unlabeled data can introduce a bias, the bias can be removed by optimizing a weighted likelihood function where the unlabeled data is raised to the power $\frac{N_L}{N}$, where N_L is the number of labeled samples and N is the number of all samples. This corresponds to the weighting scheme used in Morfessor when setting the ratio $\frac{\alpha}{\beta} = \frac{N_L}{N}$.

Adaptor Grammars. Recently, Sirts and Goldwater (2013) presented work on minimally supervised morphological segmentation using the adaptor grammar (AG) approach (Johnson, Griffiths, and Goldwater 2006). The AGs are a non-parametric Bayesian modeling framework applicable for learning latent tree structures over an input corpus of strings. They can be used to define morphological grammars of different complexity, starting from the simplest grammar where each word is just a sequence of morphs and extending to more complex grammars, where each word consists, for example, of zero or more prefixes, a stem, and zero or more suffixes.

The actual forms of the morphs are learned from the data and, subsequent to learning, used to generate segmentations for new word forms. In this general approach, AGs are similar to the Morfessor family (Creutz and Lagus 2007). A major difference, however, is that the morphological grammar is not hard-coded but instead specified as an input to the algorithm. This allows different grammars to be explored in a flexible manner. Prior to the work by Sirts and Goldwater, the AGs were successfully applied in a related task of segmenting utterances into words (Johnson 2008; Johnson and Goldwater 2009; Johnson and Demuth 2010).

The second major difference between the Morfessor family and the AG framework is the contrast between the MAP and fully Bayesian estimation approaches. Whereas the search procedure of the Morfessor method discussed earlier returns a single model corresponding to the MAP point-estimate, AGs instead operate with full posterior distributions over all possible models. Because acquiring the posteriors analytically is intractable, **inference** is performed utilizing Markov chain Monte Carlo algorithms to obtain samples from the posterior distributions of interest (Johnson 2008; Johnson and Goldwater 2009; Johnson and Demuth 2010; Sirts and Goldwater 2013). However, as sampling-based models are costly to train on large amounts of data, we adopt the parsing-based method proposed in Sirts and Goldwater (2013) to use the trained AG model inductively on test data. One of the byproducts of training the AG model is the **posterior grammar**, which in addition to all the initial grammar rules, also contains the cached subtrees learned by the system. This grammar can be used in any standard parser to obtain segmentations for new data.

The AG framework was originally designed for the unsupervised learning setting, but Sirts and Goldwater (2013) introduced two approaches for semi-supervised learning they call the semi-supervised AG and AG Select methods. The semi-supervised AG approach is an extension to unsupervised AG, in which the annotated data $\mathcal{D}$ is exploited in a straightforward manner by keeping the annotated parts of parse trees fixed while inferring latent structures for the unannotated parts. For unannotated word forms, inference is performed on full trees. For example, the grammar may specify that words are sequences of morphs and each morph is a sequence of submorphs. Typically, the annotated data only contain morpheme boundaries and submorphs are latent in this context. In this situation the inference for annotated data is performed over submorph structures only.

Similarly to unsupervised learning, semi-supervised AG requires the morphological grammar to be defined manually. Meanwhile, the AG Select approach aims to automate the grammar development process by systematically evaluating a range of grammars and finding the best one. AG Select is trained using unsupervised AG with an uninformative **metagrammar** so that the resulting parse-trees contain many possible segmentation templates. To find out which template works the best for any given language or data set, each of these templates are evaluated using the annotated data set $\mathcal{D}$. In this sense, AG Select can be characterized as more of a model selection method than semi-supervised learning.

Conditional Random Fields. The Morfessor and AG algorithms discussed earlier, although different in several respects, operate in a similar manner in that they both learn lexicons. For Morfessor, the lexicon consists of morphs, whereas for AG, the lexical units are partial parse-trees. Subsequent to learning, new word forms are segmented either by generating the most likely morph sequences (Morfessor) or by sampling parse trees from the posterior distribution (AG). In what follows, we consider a different approach to segmentation using sequence labeling methodology. The key idea in this approach is to focus the modeling effort to **morph boundaries** instead of the whole segments. Following the presentation of Ruokolainen et al. (2013, 2014), the morphological segmentation task can be represented as a sequence labeling problem by assigning each character in a word form to one of three classes, namely,

B	beginning of a multi-character morph
M	middle of a multi-character morph
S	single-character morph

Using this label set, one can represent the segmentation of the Finnish word *autoilta* (*from cars*) (*auto+i+lta*) as

a	u	t	o	i	l	t	a
↓	↓	↓	↓	↓	↓	↓	↓
B	M	M	M	S	B	M	M

Naturally, one can also use other label sets. Essentially, by defining more fine-grained labels, one captures increasingly eloquent structure but begins to overfit model to the training data because of increasingly sparser statistics. Subsequent to defining the label set, one can learn a segmentation model using general sequence labeling methods, such as the well-known conditional random field (CRF) framework (Lafferty, McCallum, and Pereira 2001).

Denoting the word form and the corresponding label sequence as x and y, respectively, the CRFs directly model the conditional probability of the segmentation given the word form, that is, $p(y \mid x; w)$. The model parameters w are estimated discriminatively from the annotated data set $\mathcal{D}$ using iterative learning algorithms (Lafferty, McCallum, and Pereira 2001; Collins 2002). Subsequent to estimation, the CRF model segments word forms x by using maximum a posteriori (MAP) graph inference, that is, solving an optimization problem

$$z = \arg\max_{u} p\left(u \mid x; w\right) \tag{3}$$

using the standard Viterbi search (Lafferty, McCallum, and Pereira 2001).

As it turns out, the CRF model can learn to segment words with a surprisingly high accuracy from a relatively small $\mathcal{D}$, that is, without utilizing any of the available unannotated word forms $\mathcal{U}$. Particularly, Ruokolainen et al. (2013) showed that it is sufficient to use simple left and right substring context features that are naturally accommodated by the discriminative parameter estimation procedure. Moreover, Ruokolainen et al. (2014) showed that the CRF-based approach can be successfully extended to semi-supervised learning settings in a straightforward manner via feature set expansion by utilizing predictions of unsupervised segmentation algorithms. By utilizing this approach, the CRF model learns to associate the output of the unsupervised algorithms, such as the Morfessor and adaptor grammar methods, in relation to the surrounding substring context.

Other Work. In addition to the algorithms discussed here, there exist numerous other segmentation approaches applicable in the minimally supervised learning setting. As the earliest example of work in this line, consider obtaining segmentations using the classic letter successor variety (LSV) method of Harris (1955). The LSV method utilizes the insight that the predictability of successive letters should be high within morph segments, and low at the boundaries. Consequently, a high variety of letters following a prefix indicates a high probability of a boundary. Whereas LSV score tracks predictability given prefixes, the same idea can be utilized for suffixes, providing the letter predecessor variety (LPV) method. As for the minimally supervised learning setting, the LSV/LPV method can be used most straightforwardly by counting the LSV/LPV scores from unannotated data and, subsequently, tuning the necessary threshold values on the annotated data (Çöltekin 2010). On the other hand, one could also use the LSV/LPV values as features for a classification model, in which case the threshold values can be learned discriminatively based on the available annotated data. The latter approach is essentially realized in the event the LSV/PSV scores are provided for the CRF model discussed earlier (Ruokolainen et al. 2014).

As for more recent work, we first refer to the generative log-linear model of Poon, Cherry, and Toutanova (2009). Similarly to the Morfessor model family, this approach is based on defining a joint probability distribution over the unannotated word forms $\mathcal{U}$ and the corresponding segmentations $\mathcal{S}$. The distribution is log-linear in form and is denoted as $p(\mathcal{U}, \mathcal{S}; \theta)$, where θ is the model parameter vector. Again, similarly to the Morfessor framework, Poon, Cherry, and Toutanova (2009) learn a morph lexicon that is subsequently used to generate segmentations for new word forms. The learning is controlled using prior distributions on both corpus and lexicon, which penalize exceedingly complex morph lexicon (similarly to Morfessor) and exceedingly segmented corpus, respectively. The log-linear form of $p(\mathcal{U}, \mathcal{S}; \theta)$ enables the approach to use a wide range of overlapping features. Particularly, Poon, Cherry, and Toutanova (2009) utilize a morph-context feature set with individual features defined for each morph and morph substring contexts. In addition to unsupervised learning, they present experiments in the semi-supervised setting. Specifically, they accomplish this by fixing the segmentations of annotated words in $\mathcal{D}$, according to their gold standard segmentation. Note, however, that this approach of extending a generative model does not necessarily utilize the supervision efficiently, as discussed previously regarding the Morfessor method family.

Finally, we briefly mention a range of recently published methods (Monson, Hollingshead, and Roark 2010; Spiegler and Flach 2010; Kılıç and Bozşahin 2012; Eger 2013). The Paramor approach presented by Monson, Hollingshead, and Roark (2010) defines a rule-based system for unsupervised learning of morphological paradigms. The Promodes system of Spiegler and Flach (2010) defines a family of generative probabilistic models for recovering segment boundaries in an unsupervised fashion. The algorithm of Kılıç and Bozşahin (2012) is based on a generative hidden Markov model (HMM), in which the HMM learns to generate morph sequences for given word forms in a semi-supervised fashion. Finally, Eger (2013) presents work on fully supervised segmentation by exhaustive enumeration and a generative Markov model on morphs. As for the minimally supervised learning setting, the Paramor system learns mainly from unannotated data $\mathcal{U}$ and utilizes annotated data $\mathcal{D}$ to adjust the required threshold value. The Promodes models can be trained either in an unsupervised manner on $\mathcal{U}$ or in a supervised manner on $\mathcal{D}$. The algorithm of Kılıç and Bozşahin (2012) learns mainly from unannotated data $\mathcal{U}$ and incorporates supervision from the annotated corpus in the form of manually selected statistics: the inclusion of the statistics yields a large

improvement in performance. Lastly, in their work with the supervised enumeration approach, Eger (2013) assumes a large (on the order of tens of thousands) amount of annotated word forms available for learning. Thus, it is left for future work to determine if the approach could be applied successfully in the minimally supervised learning setting.

3.3.2 Summary. Here we aim to summarize the fundamental differences and similarities between the varying learning approaches discussed in the previous section.

Learning Lexicons versus Detecting Boundaries. We begin by dividing the methods described earlier into two—**lexicon-based** (Creutz et al. 2007; Poon, Cherry, and Toutanova 2009; Monson, Hollingshead, and Roark 2010; Kılıç and Bozşahin 2012; Eger 2013; Sirts and Goldwater 2013) and **boundary detection** (Harris 1955; Spiegler and Flach 2010; Ruokolainen et al. 2013)—categories. In the former, the model learns lexical units, whereas in the latter the model learns properties of morph boundaries. For example, in the case of Morfessor (Creutz et al. 2007) the lexical units correspond to morphs whereas in AGs (Sirts and Goldwater 2013) the units are parse trees. Meanwhile, consider the CRF approach of Ruokolainen et al. (2013) and the classical approach of Harris (1955), which identify morph boundary positions using substring contexts and letter successor varieties, respectively. In general, whether it is easier to discover morphs or morph boundaries is largely an empirical question. So far, only the method of Poon, Cherry, and Toutanova (2009) has explicitly modeled both a morph lexicon and features describing character n-grams at morpheme boundaries.

Generative versus Discriminative Learning. The second main distinction divides the models into **generative** and **discriminative** approaches. The generative approaches (Creutz et al. 2007; Poon, Cherry, and Toutanova 2009; Spiegler and Flach 2010; Monson, Hollingshead, and Roark 2010; Kılıç and Bozşahin 2012; Eger 2013; Sirts and Goldwater 2013) model the joint distribution of word forms and their corresponding segmentations, whereas discriminative (Harris 1955; Ruokolainen et al. 2013) approaches directly estimate a conditional distribution of segmentation *given* a word form. In other words, whereas generative methods generate both word forms and segmentations, the discriminative methods generate only segmentations given word forms. The generative models are naturally applicable for unsupervised learning. Meanwhile, discriminative modeling always requires some annotated data, thus excluding the possibility of unsupervised learning. Lastly, it appears that most lexicon-based methods are generative and most boundary detection methods are discriminative. However, note that this is a trend rather than a rule, as exemplified by generative boundary detection method of Spiegler and Flach (2010).

Semi-Supervised Learning Approaches. Both generative and discriminative models can be extended to utilize annotated as well as unannotated data in a semi-supervised manner. However, the applicable techniques differ. For generative models, semi-supervised learning is in principle trivial: For the labeled word forms $\mathcal{D}$, the segmentation is fixed to its correct value, as exemplified by the approaches of Poon, Cherry, and Toutanova (2009), Spiegler and Flach (2010), and Sirts and Goldwater (2013). On the other hand, the semi-supervised setting also makes it possible to apply discriminative techniques to generative models. In particular, model hyperparameters can be selected to optimize segmentation performance, rather than some generative objective, such as likelihood. Special cases of hyperparameter selection include the weighted objective function

(Kohonen, Virpioja, and Lagus 2010), data selection (Virpioja, Kohonen, and Lagus 2011; Sirts and Goldwater 2013), and grammar template selection (Sirts and Goldwater 2013). As for the weighted objective function and grammar template selection, the weights and templates are optimized to maximize segmentation accuracy. Meanwhile, data selection is based on the observation that omitting some of the training data can improve segmentation accuracy (Virpioja, Kohonen, and Lagus 2011; Sirts and Goldwater 2013).

For discriminative models, the possibly most straightforward semi-supervised learning technique is adding features derived from the unlabeled data, as exemplified by the CRF approach of Ruokolainen et al. (2014). However, discriminative, semi-supervised learning is in general a much researched field with numerous diverse techniques (Zhu and Goldberg 2009). For example, merely for the CRF model alone, there exist several proposed semi-supervised learning approaches (Jiao et al. 2006; Mann and McCallum 2008; Wang et al. 2009).

On Local Search. In what follows, we will discuss a potential pitfall of some algorithms that utilize local search procedures in the parameter estimation process, as exemplified by the Morfessor model family (Creutz et al. 2007). As discussed in Section 3.3.1, the Morfessor algorithm finds a local optimum of the objective function using a local search procedure. This complicates model development because if two model variants perform differently empirically, it is uncertain whether it is because of a truly better model or merely better fit with the utilized parameter estimation method, as discussed also by Goldwater (2006, Section 4.2.2.3). Therefore, in contrast, within the adaptor grammar framework (Johnson, Griffiths, and Goldwater 2006; Sirts and Goldwater 2013), the focus has not been on finding a single best model, but rather on finding the posterior distribution over segmentations of the words. Another approach to the problem of bad local optima is to start a local search near some known good solution. This approach is taken in Morfessor FlatCat, for which it was found that initializing the model with the segmentations produced by the supervised CRF model (with a convex objective function) yields improved results (Grönroos et al. 2014).

4. Experiments

In this section, we perform an empirical comparison of segmentation algorithms in the semi-supervised learning setting. The purpose of the presented experiments is to extend the current literature by considering a wider range of languages compared with previous work, and by providing an in-depth error analysis.

4.1 Data

We perform the experiments on four languages, namely, English, Estonian, Finnish, and Turkish. The English, Finnish, and Turkish data are from the Morpho Challenge 2009/2010 data set (Kurimo et al. 2009; Kurimo, Virpioja, and Turunen 2010). The annotated Estonian data set is acquired from a manually annotated, morphologically disambiguated corpus,[1] and the unannotated word forms are gathered from the Estonian Reference Corpus (Kaalep et al. 2010). Table 3 shows the total number of instances available for model estimation and testing.

1 Available at `http://www.cl.ut.ee/korpused/morfkorpus/index.php?lang=en`.

Table 3
Number of word types in the data sets.

	English	Estonian	Finnish	Turkish
train (unannotated)	384,903	3,908,820	2,206,719	617,298
train (annotated)	1,000	1,000	1,000	1,000
development	694	800	835	763
test	10×1,000	10×1,000	10×1,000	10×1,000

4.2 Compared Algorithms

We present a comparison of the Morfessor family (Creutz and Lagus 2002, 2005, 2007; Kohonen, Virpioja, and Lagus 2010; Grönroos et al. 2014), the adaptor grammar framework (Sirts and Goldwater 2013), and the conditional random fields (Ruokolainen et al. 2013, 2014). These methods have freely available implementations for research purposes.

The log-linear model presented by Poon, Cherry, and Toutanova (2009) is omitted because it does not have a freely available implementation. However, the model has been compared in the semi-supervised learning setting on Arabic and Hebrew with CRFs and Morfessor previously by Ruokolainen et al. (2013). In these experiments, the model was substantially outperformed on both languages by the CRF method and on Hebrew by Morfessor.

In order to provide a strong baseline for unsupervised learning results, we performed preliminary experiments using the model presented by Lee, Haghighi, and Barzilay (2011).[2] Their model learns segmentation in an unsupervised manner by exploiting syntactic context of word forms observed in running text and has shown promising results for segmentation of Arabic. In practice, we found that when using the method's default hyperparameters, it did not yield nearly as good results as the other unsupervised methods on our studied data sets. Adjusting the hyperparameters turns out to be complicated by the computational demands of the method. When utilizing the same computer set-up as for the other models, training the method requires limiting the maximum word length of analyzed words to 12 in order for the model to fit in memory, as well as requiring weeks of runtime for a single run. We decided to abandon further experimentation with the method of Lee, Haghighi, and Barzilay (2011), as optimizing its hyperparameters was computationally infeasible.

4.3 Evaluation

This section describes the utilized evaluation measures and the performed error analysis.

4.3.1 Boundary Precision, Recall, and F1-score. The word segmentations are evaluated by comparison with reference segmentations using **boundary precision**, **boundary recall**, and **boundary F1-score**. The boundary F1-score, or F1-score for short, equals the harmonic mean of precision (the percentage of correctly assigned boundaries with respect

2 Implementation is available at `http://people.csail.mit.edu/yklee/code.html`.

to all assigned boundaries) and recall (the percentage of correctly assigned boundaries with respect to the reference boundaries):

$$\text{Precision} = \frac{C(\text{correct})}{C(\text{proposed})}, \tag{4}$$

$$\text{Recall} = \frac{C(\text{correct})}{C(\text{reference})}. \tag{5}$$

We follow Virpioja et al. (2011) and use type-based macro-averages. However, we handle word forms with alternative analyses in a different fashion. Instead of penalizing algorithms that propose an incorrect number of alternative analyses, we take the best match over the alternative reference analyses (separately for precision and recall). This is because all the methods considered in the experiments provide a single segmentation per word form.

Throughout the experiments, we establish statistical significance with confidence level 0.95, according to the standard one-sided Wilcoxon signed-rank test performed on 10 random subsets of 1,000 word forms drawn from the complete test sets (subsets may contain overlapping word forms).

Because we apply a different treatment of alternative analyses, the results reported in this article are not directly comparable to the boundary F1-scores reported for the Morpho Challenge competitions (Kurimo et al. 2009; Kurimo, Virpioja, and Turunen 2010). However, the best boundary F1-scores for all languages reported in Morpho Challenge have been achieved with the semi-supervised Morfessor Baseline algorithm (Kohonen, Virpioja, and Lagus 2010), which is included in the current experiments.

4.3.2 Error Analysis. We next discuss the performed error analysis. The purpose of the error analysis is to gain a more detailed understanding into what kind of errors the methods make, and how the error types affect the overall F1-scores. To this end, we use a categorization of morphs into the categories PREFIX, STEM, and SUFFIX, in addition defining a separate category for DASH. For the English and Finnish sections of the Morpho Challenge data set, the segmentation gold standard annotation contain additional information for each morph, such as part-of-speech for stems and morphological categories for affixes, which allows us to assign each morph into one of the morph type categories. In some rare cases the tagging is not specific enough, and we choose to assign the tag UNKNOWN. However, as we are evaluating segmentations, we lack the morph category information for the proposed analyses. Consequently, we cannot apply a straightforward category evaluation metric, such as category F1-score. In what follows, we instead show how to use the categorization on the gold standard side to characterize the segmentation errors.

We first observe that errors come in two kinds, **over-segmentation** and **under-segmentation**. In over-segmentation, boundaries are incorrectly assigned within morph segments, and in under-segmentation, the segmentation fails to uncover correct morph boundaries. For example, consider the English compound word form *girlfriend* with a correct analysis *girl+friend*. Then, an under-segmentation error occurs in the event the model fails to assign a boundary between the segments *girl* and *friend*. Meanwhile, over-segmentation errors take place if any boundaries are assigned within the two compound segments *girl* and *friend*, such as *g+irl* or *fri+end*.

As for the relationship between these two error types and the precision and recall measures in Equations (4) and (5), we note that over-segmentation solely affects

precision, whereas under-segmentation only affects recall. This is evident as the measures can be written equivalently as:

$$\text{Precision} = \frac{C(\text{proposed}) - C(\text{over-segm.})}{C(\text{proposed})} = 1 - \frac{C(\text{over-segm.})}{C(\text{proposed})}, \tag{6}$$

$$\text{Recall} = \frac{C(\text{reference}) - C(\text{under-segm.})}{C(\text{reference})} = 1 - \frac{C(\text{under-segm.})}{C(\text{reference})}. \tag{7}$$

In the error analysis, we use these equivalent expressions as they allow us to examine the effect of *reduction* in precision and recall caused by over-segmentation and under-segmentation, respectively.

The over-segmentation errors occur when a segment that should remain intact is split. Thus, these errors can be assigned into categories c according to the morph tags PREFIX, STEM, SUFFIX, and UNKNOWN. The segments in the category DASH cannot be segmented and do not, therefore, contribute to over-segmentation errors. We then decompose the precision and recall reductions in Equations (6) and (7) into those caused by errors in each category indexed by c and d:

$$\text{Precision} = 1 - \sum_c \frac{C(\text{over-segm. } (c))}{C(\text{proposed})}, \tag{8}$$

$$\text{Recall} = 1 - \sum_d \frac{C(\text{under-segm. } (d))}{C(\text{reference})}. \tag{9}$$

Equation (8) holds because

$$\frac{C(\text{over-segm.})}{C(\text{reference})} = \frac{\sum_c C(\text{over-segm. } (c))}{C(\text{reference})} = \sum_c \frac{C(\text{over-segm. } (c))}{C(\text{reference})}, \tag{10}$$

where c indexes the over-segmentation error categories. The expression for recall in Equation (9) can be derived analogously, but it must be noted that the categorization d by error type differs from that of precision as each under-segmentation error occurs at a segment boundary, such as STEM-SUFFIX, STEM-STEM, PREFIX-STEM, rather than in the middle of a segment. To simplify analysis, we have grouped all segment boundaries, in which either the left or right segment category is DASH into the CONTAINS DASH category. Boundary types that occur fewer than 100 times in the test data are merged into the OTHER category.

Table 4 shows the occurrence frequency of each boundary category, averaged over alternative analyses. Evidently, we expect the total precision scores to be most influenced by over-segmentation of STEM and SUFFIX segment types because of their high frequencies. Similarly, the overall recall scores are expected to be most impacted by under-segmentation of STEM-SUFFIX and SUFFIX-SUFFIX boundaries. Finnish is also substantially influenced by the STEM-STEM boundary, indicating that Finnish uses compounding frequently.

For simplicity, when calculating the error analysis, we forgo the sampling procedure of taking $10 \times 1,000$ word forms from the test set, used for the overall F1-score for statistical significance testing by Virpioja et al. (2011). Rather, we calculate the error analysis on the union of these sampled sets. As the sampling procedure may introduce

Table 4
Absolute and relative frequencies of the boundary categories in the error analysis. The numbers are averaged over the alternative analyses in the reference annotation.

Category	English		Finnish	
STEM	38,608.8	(82.2%)	72,666.0	(81.3%)
SUFFIX	7,172.9	(15.3%)	15,384.9	(17.2%)
PREFIX	1,152.8	(2.5%)	946.5	(1.1%)
UNKNOWN	54.5	(0.1%)	414.0	(0.5%)
STEM-SUFFIX	5,349.2	(62.6%)	9,889.9	(45.8%)
SUFFIX-SUFFIX	1,481.0	(17.3%)	5,917.5	(27.4%)
STEM-STEM	613.4	(7.2%)	3,538.0	(16.4%)
SUFFIX-STEM	n/a	n/a	1,501.0	(6.9%)
CONTAINS DASH	458.0	(6.5%)	426.0	(2.0%)
PREFIX-STEM	554.3	(5.4%)	235.2	(1.1%)
OTHER	91.0	(1.1%)	105.4	(0.5%)

the same word form in several samples, the error analysis precisions and recalls are not necessarily identical to the ones reported for the overall results.

In summary, although we cannot apply category F1-scores, we can instead categorize each error by type. These categories then map directly to either reduced precision or recall. Interpreting precision and recall requires some care as it is always possible to reduce over-segmentation errors by segmenting less and, conversely, to reduce under-segmentation errors by segmenting more. However, if this is taken into account, the error categorization can be quite informative.

4.4 Model Learning and Implementation Specifics

4.4.1 Morfessor. We use a recently released Python implementation of the Morfessor method (Virpioja et al. 2013; Smit et al. 2014).[3] The package implements both the unsupervised and semi-supervised Morfessor Baseline (Creutz and Lagus 2002, 2007; Kohonen, Virpioja, and Lagus 2010). For Morfessor FlatCat we apply the Python implementation by Grönroos et al. (2014).[4]

In its original formulation, the unsupervised Morfessor Baseline uses no hyperparameters. However, it was found by Virpioja, Kohonen, and Lagus (2011) that performance does not improve consitently with growing data because the method segments less on average for each added training word form. Therefore, we optimize the training data size by including only the most frequent words in the following sizes: 10k, 20k, 30k, 40k, 50k, 100k, 200k, 400k, ..., as well as the full set. We then choose the model yielding highest F1-score on the development set.

As for semi-supervised training of Morfessor Baseline, we perform a grid search on the development set for the hyperparameter β (see Section 3.3.1). For each value of β we use the automatic adaptation of the hyperparameter α provided by the implementation. The automatic adaptation procedure is applied during model training and is, therefore, computationally less demanding compared with grid search. Intuitively, the adaptation

3 Available at `https://github.com/aalto-speech/morfessor`.
4 Available at `https://github.com/aalto-speech/flatcat`.

functions as follows. The hyperparameter α affects how much the method segments on average. Although optimizing it for segmentation performance during training is non-trivial, one can instead apply the heuristic that the method should neither over-segment nor under-segment. Therefore, the implementation adjusts α such that the development set precision and recall become approximately equal.

In the semi-supervised training for Morfessor FlatCat, the segmentations are initialized to the ones produced by the supervised CRF model trained with the same amount of labeled training data. As automatic adaptation of the hyperparameter α has not yet been implemented for Morfessor FlatCat, values for both α and β are found by a combined grid search on the development set. The computational demands of the grid search were reduced by using the optimal hyperparameter values for Morfessor Baseline as an initial guess when constructing the grid. We also choose the non-morpheme removal heuristics used by Morfessor FlatCat for each language separately using the development set. For English, Estonian, and Finnish the heuristics described by Grönroos et al. (2014) are beneficial, but they do not fit Turkish morphology as well. For Turkish we convert non-morphemes into suffixes or stems, without modifying the segmentation.

4.4.2 Adaptor Grammars. The technical details of the AG model are described by Johnson, Griffiths, and Goldwater (2006) and the inference details are described by Johnson, Griffiths, and Goldwater (2007). For unsupervised AG learning, we used the freely available implementation,[5] which was also the basis for the semi-supervised implementation. Table label resampling was turned on and all hyperparameters were inferred automatically as described by Johnson and Goldwater (2009). The metagrammar for AG Select is the same as described by Sirts and Goldwater (2013). Inductive learning with the posterior grammar was done with a freely available CKY parser.[6] For both unsupervised and semisupervised AG, we use a three-level collocation-submorph grammar in which the final segmentation is parsed out as a sequence of Morphs:

$$\begin{aligned} \text{Word} &\rightarrow \underline{\text{Colloc}}^{+} \\ \underline{\text{Colloc}} &\rightarrow \underline{\text{Morph}}^{+} \\ \underline{\text{Morph}} &\rightarrow \underline{\text{SubMorph}}^{+} \\ \underline{\text{SubMorph}} &\rightarrow \text{Char}^{+} \end{aligned}$$

We experimented with two types of grammars, where the Word non-terminal is either cached or not. These two grammar versions have no difference when trained transductively. However, when training an inductive model, it may be beneficial to store the subtrees corresponding to whole words because these trees can be used to parse the words in the test set that were seen during training with a single rule. All models, both unsupervised and semi-supervised, are trained on 50k most frequent word types. For semi-supervised experiments, we upweight the labeled data by an integer number of times by repeatedly caching the subtrees corresponding to morphemes in the annotated data. The additional cached subtrees are rooted in the Morph non-terminal. Similarly to semi-supervised Morfessor, we experimented with initializing the segmentations with

5 Available at `http://web.science.mq.edu.au/~mjohnson/Software.htm`.
6 Also obtained from `http://web.science.mq.edu.au/~mjohnson/Software.htm`.

the output of the supervised CRF model, which in some cases resulted in improved accuracy over the random initialization. We searched the optimal values for each experiment for the upweighting factor, cached versus non-cached root non-terminal, and random versus CRF initialization on the development set.

An AG model is stochastic and each segmentation result is just a single sample from the posterior. A common approach in such a case is to take several samples and report the average result. Maximum marginal decoding (MMD) (Johnson and Goldwater 2009; Stallard et al. 2012) that constructs a marginal distribution from several independent samples and returns their mean value has been shown to improve the sampling-based models' results about 1–2 percentage points. Although the AG model uses sampling for training, the MMD is not applicable here because during test time the segmentations are obtained using parsing. However, we propose another way of achieving the gain in a similar range to the MMD. We train five different models and concatenate their posterior grammars into a single joint grammar, which is then used as the final model to decode the test data. Our experiments show that the posterior grammar concatentation, similarly to the MMD, leads to consistent improvements of 1–2 percentage points over the mean of the individual samples.

4.4.3 CRFs. The utilized Python implementation of the CRF model follows the presentation of Ruokolainen et al. (2013, 2014).[7] As for the left and right substring features incorporated in the model, we include all substrings that occur in the training data. The maximum substring length and averaged perceptron learning of CRF model parameters are optimized on the held-out development sets following Ruokolainen et al. (2013). For semi-supervised learning, we utilize log-normalized successor and predecessor variety scores and binary Morfessor Baseline and AG features following the presentation of Ruokolainen et al. (2014). The unsupervised Morfessor Baseline and AG models are optimized on the development set as described earlier. The successor and predecessor variety scores are estimated from all the available unannotated word forms apart from words with a corpus frequency of one. The count cutoff is applied as a means of noise reduction by removing peripheral phenomena, such as misspellings.

4.5 Results

Here we summarize the results obtained using our experiment set-up. We present overall segmentation accuracies and error analysis in Sections 4.5.1 and 4.5.2, respectively. We then discuss the results in Section 4.6.

4.5.1 Boundary Precisions, Recalls, and F1-scores. In what follows, we first review unsupervised and supervised results and, subsequently, assess the semi-supervised results.

Segmentation accuracies using unsupervised and supervised methods are presented in Table 5. As for the supervised learning using the CRF model, we report segmentation accuracies obtained using 100 and 1,000 annotated word forms. Evidently, utilizing annotated data provides a distinct advantage over learning from unannotated data. Particularly, learning the supervised CRFs using 1,000 annotated word forms results in substantially higher segmentation accuracies compared with learning in an unsupervised manner from hundreds of thousands or millions of word forms. In fact, using merely 100 annotated instances results in higher accuracies in English and Turkish

7 Available at `http://users.ics.aalto.fi/tpruokol/`.

Table 5
Precision, recall, and F1-scores for unsupervised and supervised methods.

Method	Train (ann.)	Train (unann.)	Pre.	Rec.	F1
English					
MORFESSOR BASELINE (USV)	0	384,903	76.3	76.3	76.3
AG (USV)	0	384,903	62.2	84.4	71.7
CRF (SV)	100	0	86.0	72.7	78.8
CRF (SV)	1,000	0	91.6	81.2	86.1
Estonian					
MORFESSOR BASELINE (USV)	0	3,908,820	76.4	70.4	73.3
AG (USV)	0	3,908,820	78.4	73.4	75.8
CRF (SV)	100	0	79.2	59.1	67.7
CRF (SV)	1,000	0	88.4	76.7	82.1
Finnish					
MORFESSOR BASELINE (USV)	0	2,206,719	70.2	51.9	59.7
AG (USV)	0	2,206,719	68.1	68.1	68.1
CRF (SV)	100	0	73.0	59.4	65.5
CRF (SV)	1,000	0	88.3	79.7	83.8
Turkish					
MORFESSOR BASELINE (USV)	0	617,298	67.9	65.8	66.8
AG (USV)	0	617,298	72.7	76.5	74.6
CRF (SV)	100	0	84.6	71.8	77.7
CRF (SV)	1,000	0	90.0	87.3	88.6

The columns titled *Train (unann.)* denote the number of unannotated word forms utilized in learning. The columns titled *Train (ann.)* denote the number of annotated word forms.

compared with the unsupervised methods. The balance between precision and recall can be analyzed to assess how well the different methods are tuned to the amount of segmentation present in the gold standard. As discussed in Section 4.3.2, high precision in combination with low recall indicates under-segmentation, whereas high recall and low precision indicates over-segmentation. Morfessor appears to favor precision over recall (see Finnish) in the event a trade-off takes place. In contrast, the AG heavily favors recall (see English). Meanwhile, the supervised CRF model consistently prefers higher precision over recall.

These unsupervised and supervised learning results utilize the available data only partially. Thus, we next discuss results obtained using semi-supervised learning—that is, when utilizing all available annotated and unannotated word forms. The obtained segmentation accuracies are presented in Table 6. We summarize the results as follows. First, the semi-supervised CRF approach CRF (SSV) yielded highest segmentation accuracies for all considered languages and data set sizes. The improvements over other models are statistically significant. Compared with the supervised CRF model, the semi-supervised extension successfully increases the recall while maintaining the high precision. As for the Morfessor family, MORF.FC (SSV) yields significantly higher F1-scores compared with MORF.BL (SSV) on all languages. However, we found that without the CRF initialization of MORF.FC (SSV), the performance gap decreases substantially (cf. similar results reported by Grönroos et al. [2014]). On the other hand, the variants appear to behave in a similar manner in that, in the majority of cases, both approaches increase the obtained precision and recall in a balanced manner compared with the unsupervised approach MORF. BL (USV). Meanwhile, the AG variants AG (SSV) and

Table 6
Precision, recall, and F1-scores for semi-supervised methods.

Method	Train (ann.)	Train (unann.)	Pre.	Rec.	F1
English					
MORFESSOR BASELINE (SSV)	100	384,903	81.7	82.8	82.2
MORFESSOR FLATCAT (SSV)	100	384,903	83.6	83.0	83.3
AG (SSV)	100	384,903	69.0	85.8	76.5
AG SELECT (SSV)	100	384,903	75.9	79.4	77.6
CRF (SSV)	100	384,903	87.6	81.0	**84.2**
MORFESSOR BASELINE (SSV)	1,000	384,903	84.4	83.9	84.1
MORFESSOR FLATCAT (SSV)	1,000	384,903	86.9	85.2	86.0
AG (SSV)	1,000	384,903	69.8	87.1	77.5
AG SELECT (SSV)	1,000	384,903	76.7	82.3	79.4
CRF (SSV)	1,000	384,903	89.3	87.0	**88.1**
Estonian					
MORFESSOR BASELINE (SSV)	100	3,908,820	77.0	76.1	76.5
MORFESSOR FLATCAT (SSV)	100	3,908,820	81.8	74.5	77.9
AG (SSV)	100	3,908,820	71.8	75.5	73.6
AG SELECT (SSV)	100	3,908,820	60.9	90.4	72.8
CRF (SSV)	100	3,908,820	81.5	82.1	**81.8**
MORFESSOR BASELINE (SSV)	1,000	3,908,820	80.6	80.7	80.7
MORFESSOR FLATCAT (SSV)	1,000	3,908,820	84.7	82.0	83.3
AG (SSV)	1,000	3,908,820	67.1	88.8	76.4
AG SELECT (SSV)	1,000	3,908,820	62.8	90.3	74.1
CRF (SSV)	1,000	3,908,820	90.2	86.3	**88.2**
Finnish					
MORFESSOR BASELINE (SSV)	100	2,206,719	69.8	70.8	70.3
MORFESSOR FLATCAT (SSV)	100	2,206,719	77.6	73.6	75.5
AG (SSV)	100	2,206,719	65.5	70.5	67.9
AG SELECT (SSV)	100	2,206,719	66.8	73.6	70.0
CRF (SSV)	100	2,206,719	80.0	77.4	**78.7**
MORFESSOR BASELINE (SSV)	1,000	2,206,719	76.0	78.0	77.0
MORFESSOR FLATCAT (SSV)	1,000	2,206,719	81.6	80.2	80.9
AG (SSV)	1,000	2,206,719	69.7	77.6	73.4
AG SELECT (SSV)	1,000	2,206,719	69.4	74.3	71.8
CRF (SSV)	1,000	2,206,719	89.3	87.9	**88.6**
Turkish					
MORFESSOR BASELINE (SSV)	100	617,298	76.6	80.5	78.5
MORFESSOR FLATCAT (SSV)	100	617,298	80.2	83.9	82.0
AG (SSV)	100	617,298	74.1	82.8	78.2
AG SELECT (SSV)	100	617,298	69.0	82.3	75.0
CRF (SSV)	100	617,298	81.3	86.0	**83.5**
MORFESSOR BASELINE (SSV)	1,000	617,298	85.1	89.4	87.2
MORFESSOR FLATCAT (SSV)	1,000	617,298	84.9	92.2	88.4
AG (SSV)	1,000	617,298	77.0	90.9	83.4
AG SELECT (SSV)	1,000	617,298	70.5	80.4	75.1
CRF (SSV)	1,000	617,298	89.3	92.0	**90.7**

AG SELECT (SSV) heavily favor recall over precision, indicating over-segmentation.[8] Lastly, in contrast with the unsupervised learning results, in the semi-supervised setting the AG framework is significantly outperformed by the Morfessor variants.

4.5.2 Error Analysis. Next, we examine how different error types contribute to the obtained precision and recall measures, and, consequently, the overall F1-scores. To this end, we discuss the error analyses for English and Finnish presented in Tables 7 and 8, respectively.

Baselines. The first two lines in Tables 7 and 8 present the baseline models WORDS and LETTERS. The WORDS model corresponds to an approach in which no segmentation is performed, that is, all the word forms are kept intact. The LETTERS approach assigns a segment boundary between all adjacent letters. These approaches maximize precision (WORDS) and recall (LETTERS) at the cost of the other. In other words, no model can produce more over-segmentation errors compared with LETTERS, and no model can produce more under-segmentation errors compared with WORDS.[9]

Given the baseline results, we observe that the overall precision scores are most influenced by over-segmentation of STEM and SUFFIX segment types because of their high frequencies. Similarly, the overall recall scores are most impacted by under-segmentation of STEM-SUFFIX and SUFFIX-SUFFIX boundaries. Finnish recall is also substantially influenced by the STEM-STEM boundary, indicating that Finnish uses compounding frequently.

Morfessor. Similarly to the baseline (WORDS and LETTERS) results, the majority of over-segmentation errors yielded by the Morfessor variants take place within the STEM and SUFFIX segments, and most under-segmentation errors occur at the STEM-SUFFIX and SUFFIX-SUFFIX boundaries. When shifting from unsupervised learning using MORF.BL (USV) to semi-supervised learning using MORF.BL (SSV) and MORF.FC (SSV), the over-segmentation problems are alleviated rather substantially, resulting in higher overall precision scores. For example, consider the word form *countermanded*, for which MORF.BL (SSV) assigns the correct segmentation *countermand+ed*, but which is severely oversegmented by MORF.BL (USV) as *counter+man+d+ed*. One also observes a dramatic increase in the overall recall scores, indicating a smaller amount of under-segmentation taking place. For example, consider the word form *products*, for which MORF.BL (SSV) assigns the correct segmentation *product+s*, but for which MORF.BL (USV) assigns no boundaries. However, the under-segmentation errors do not decrease consistently: Although the STEM-SUFFIX and SUFFIX-SUFFIX errors are decreased substantially, one additionally observes a decline or no change in the model's ability to uncover STEM-STEM and PREFIX-STEM boundaries.

8 Generally, in the presence of annotated training data, under-segmentation and over-segmentation can be avoided by explicitly tuning the average level of segmentation. Such tuning is performed for Morfessor with the weighted objective function and for AG by choosing the level in the parse tree from which to extract the segmentations. By default, the AG segmentations were extracted from the Morph level as this gave the highest score on the development set. However, the Estonian segmentations are extracted from the Colloc level, which also explains why in the Estonian case the precision is higher than recall. These results suggest that AG (SSV) may benefit from yet another layer in the grammar, which would help to learn a better balance between precision and recall.

9 Intuitively, WORDS should yield zero recall. However, when applying macro averaging, a word having a gold standard analysis with no boundaries yields a zero denominator and is therefore undefined. To correct for this, we interpret such words as having recall 1, which explains the non-zero recall for WORDS.

Table 7
Error analysis for English.

	Over-Segmentation					Under-Segmentation						
Method	STEM	SUFFIX	PREFIX	UNKNOWN	**PRE / TOTAL**	STEM-SUFFIX	SUFFIX-SUFFIX	STEM-STEM	PREFIX-STEM	CONTAINS DASH	OTHER	**REC / TOTAL**
WORDS	0.0	0.0	0.0	0.0	100.0	55.1	8.6	5.9	4.4	2.5	0.6	23.1
LETTERS	71.1	11.8	1.7	0.3	15.1	0.0	0.0	0.0	0.0	0.0	0.0	100.0
MORF.BL (USV)	20.6	2.9	0.0	0.1	76.4	17.0	4.7	0.6	1.1	0.0	0.2	76.4
MORF.BL (SSV)	14.3	1.3	0.1	0.0	84.4	9.8	0.6	2.8	2.1	0.0	0.4	84.3
MORF.FC (SSV)	11.2	1.7	0.0	0.1	87.1	8.6	0.5	2.2	2.5	0.1	0.4	85.5
AG (USV)	31.8	5.8	0.1	0.0	62.3	10.6	3.5	0.1	0.7	0.2	0.2	84.7
AG (SSV)	27.8	2.1	0.1	0.1	70.0	10.1	1.4	0.2	0.6	0.2	0.2	87.3
AG SELECT (SSV)	18.4	4.8	0.0	0.1	76.6	8.2	1.4	2.2	4.1	1.5	0.4	82.2
CRF (SV)	7.3	0.9	0.1	0.0	91.8	10.4	0.5	4.2	2.9	0.1	0.4	81.5
CRF (SSV)	9.6	0.8	0.0	0.1	89.5	8.4	0.5	1.4	1.9	0.0	0.4	87.4

Over-segmentation and under-segmentation errors reduce precision and recall, respectively. For example, the total precision of MORF. BL (USV) is obtained as $100.0 - 20.6 - 2.9 - 0.0 - 0.1 = 76.4$. The lines MORF. BL (USV), MORF. BL (SSV), and MORF. FC (SSV) correspond to the unsupervised Morfessor Baseline, semi-supervised Morfessor Baseline, and semi-supervised Morfessor FlatCat models, respectively.

Table 8
Error analysis for Finnish.

	Over-Segmentation					Under-Segmentation							
Method	STEM	SUFFIX	PREFIX	UNKNOWN	**PRE / TOTAL**	STEM-SUFFIX	SUFFIX-SUFFIX	STEM-STEM	SUFFIX-STEM	CONTAINS DASH	PREFIX-STEM	OTHER	**REC / TOTAL**
WORDS	0.0	0.0	0.0	0.0	100.0	49.2	21.8	17.2	4.8	1.4	1.0	0.6	4.1
LETTERS	65.2	13.8	0.7	0.6	19.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
MORF.BL (USV)	26.6	3.4	0.0	0.2	69.7	28.8	17.1	1.7	0.5	0.0	0.1	0.2	51.6
MORF.BL (SSV)	20.8	2.9	0.0	0.2	76.1	13.6	5.9	1.9	0.5	0.0	0.1	0.1	78.0
MORF.FC (SSV)	15.3	2.9	0.0	0.1	81.7	12.2	5.2	1.5	0.6	0.1	0.1	0.1	80.2
AG (USV)	28.3	3.3	0.1	0.2	68.1	19.0	11.5	0.7	0.2	0.3	0.0	0.2	68.1
AG (SSV)	27.9	2.1	0.1	0.2	69.7	14.7	6.5	0.7	0.2	0.1	0.1	0.1	77.6
AG SELECT (SSV)	24.2	6.1	0.0	0.1	69.5	13.2	7.8	2.4	1.1	0.8	0.2	0.1	74.4
CRF (SV)	9.3	2.3	0.0	0.0	88.3	10.7	2.2	5.8	1.1	0.1	0.3	0.2	79.7
CRF (SSV)	9.2	1.4	0.0	0.1	89.3	8.0	2.3	1.2	0.4	0.1	0.1	0.2	87.8

Over-segmentation and under-segmentation errors reduce precision and recall, respectively. The lines MORF. BL (USV), MORF. BL (SSV), and MORF. FC (SSV) correspond to the unsupervised Morfessor Baseline, the semi-supervised Morfessor Baseline, and semi-supervised Morfessor FlatCat models, respectively.

Adaptor Grammars. Similarly to the baseline and Morfessor results, the majority of over-segmentation errors yielded by the AG variants occur within the STEM and SUFFIX segments. Compared with the unsupervised AG (USV) approach, the first semi-supervised extension AG (SSV) manages to reduce over-segmentation of the STEM segments slightly and SUFFIX segments substantially, thus resulting in overall higher precision. Meanwhile, the second extension AG SELECT (SSV) also results in overall higher precision by reducing over-segmentation of STEM segments substantially—although, for Finnish, SUFFIX is oversegmented compared with AG SELECT (USV). On the other hand, whereas both AG (SSV) and AG SELECT (SSV) improve recall on Finnish compared to AG (USV), only AG (SSV) succeeds in improving recall for English. This is because the AG SELECT (SSV) variant decreases the model's ability to capture other than STEM-SUFFIX and SUFFIX-SUFFIX boundaries compared with the unsupervised AG (USV) approach.

Conditional Random Fields. In contrast to the Morfessor and AG frameworks, the error patterns produced by the CRF approach do not directly follow the baseline approaches. Particularly, we note that the supervised CRF (SV) approach successfully captures SUFFIX-SUFFIX boundaries and fails to find STEM-STEM boundaries—that is, behaves in an opposite manner compared with the baseline results. CRF (SV) also under-segments the less-frequent PREFIX-STEM and STEM-SUFFIX boundaries for English and Finnish, respectively. Meanwhile, the semi-supervised extension CRF (SSV) alleviates the problem of finding STEM-STEM boundaries substantially, resulting in improvement in overall recall. For example, CRF (SSV) correctly segments compound forms *rainstorm* and *windpipe* as *rain+storm* and *wind+pipe*, whereas CRF (SV) incorrectly assigns no segmentation boundaries to either of these forms. Note that improving recall means that CRF (SSV) is required to segment more compared with CRF (SV). For English, this increased segmentation results in a slight increase in over-segmentation of STEM—that is, the model trades off the increase in recall for precision. For example, whereas CRF (SV) correctly segments *ledgers* as *ledger+s*, CRF (SSV) yields an incorrect segmentation *led+ger+s*.

4.6 Discussion

When increasing the amount of data utilized for learning—that is, when shifting from fully unsupervised or supervised learning to semi-supervised learning—we naturally expect the segmentation method families to improve their performance measured using the F1-score. Indeed, as shown in Tables 5 and 6, this improvement takes place within all considered approaches. In some cases, as exemplified by the CRF model on English, achieving a higher F1-score may require a trade-off between precision and recall—that is, the model lowers precision somewhat to gain recall (or vice versa). However, by examining the error analyses in Tables 7 and 8, we also observe the occurrence of a second kind of trade-off, in which the semi-supervised Morfessor and AG approaches trade off under-segmentation errors to other under-segmentation errors. Particularly, although the STEM-SUFFIX and SUFFIX-SUFFIX boundary recall errors are decreased, one also observes an increase in the errors at STEM-STEM and PREFIX-STEM boundaries. This type of behavior indicates an inherent inefficiency in the models' ability to utilize increasing amounts of data.

Next, we discuss potential explanations for the empirical success of the discriminatively trained CRF approach. First, discriminative training has the advantage of directly optimizing segmentation accuracy with few assumptions about the data generating process. Meanwhile, generative models can be expected to perform well only if the model

definition matches the data-generating process adequately. In general, discriminative approaches should generalize well under the condition that a sufficient amount of training data is available. Given the empirical results, this condition appears to be fulfilled for morphological segmentation in the minimally supervised setting. Second, the CRFs aim to detect boundary positions based on rich features describing substring contexts. Because the substrings are more frequent than lexical units, their use enables more efficient utilization of sparse data. For example, consider a training data that consists of a single labeled word form *kato+lla* (*on roof*). When segmenting an unseen word form *matolle* (*onto rug*), with the correct segmentation *mato+lle*, the CRFs can utilize the familiar left and right substrings *ato* and *ll*, respectively. In contrast, a lexicon-based model has a lexicon of two morphs {*kato, lla*}, neither of which match any substring of *matolle*.

Finally, we discuss how the varying approaches differ when learning to split affixes and compounds. To this end we first point out that, in the examined English and Finnish corpora, the suffix class is **closed** and has only a small number of morphemes compared with the **open** prefix and stem categories. In consequence, a large coverage of suffixes should be achievable already with a relatively small annotated data set. This observation is supported by the evident success of the fully supervised CRF method in learning suffix splitting for both considered languages. On the other hand, although more efficient at learning suffix splitting, the supervised CRF approach is apparently poor at detecting compound boundaries. Intuitively, learning compound splitting in a supervised manner seems infeasible because the majority of stem forms are simply not present in the available small annotated data set. Meanwhile, the semi-supervised CRF extension and the generative Morfessor and AG families, which do utilize the large unannotated word lists, capture the compound boundaries with an appealing high accuracy. This result again supports the intuition that in order to learn the open categories, one is required to utilize large amounts of word forms for learning. However, it appears that the necessary information can be extracted from unannotated word forms.

5. Future Work

In this section, we discuss our findings on potentially fruitful directions for future research.

5.1 On Improving Existing Approaches

Interestingly, the CRF-based segmentation method achieves its success using minimalistic, language-independent features with a simple, feature-based, semi-supervised learning extension. Therefore, it seems plausible that one could boost the accuracy further by designing richer, language-dependent feature extraction schemes. For example, one could potentially exploit features capturing vowel harmony present in Finnish, Estonian, and Turkish. As for semi-supervised learning, one can utilize unannotated word lists in a straightforward manner by using the feature set expansion approach as discussed by Ruokolainen et al. (2014). Similar expansion schemes for CRFs have also been successfully applied in the related tasks of Chinese word segmentation (Sun and Xu 2011; Wang et al. 2011) and chunking (Turian, Ratinov, and Bengio 2010). Nevertheless, there exist numerous other approaches proposed for semi-supervised learning of CRFs (Jiao et al. 2006; Mann and McCallum 2008; Wang et al. 2009) that could potentially provide an advantage over the feature-based, semi-supervised learning approach. Naturally, one could also examine utilizing these techniques simultaneously with the expanded feature sets.

As discussed in Section 3.3, it is possible for the generative models to utilize annotated data in a straightforward manner by fixing samples to their true values. This approach was taken by Poon, Cherry, and Toutanova (2009), Spiegler and Flach (2010), and Sirts and Goldwater (2013). On the other hand, as discussed in Section 3.3.1, for the Morfessor family the fixing approach was outperformed by the weighted objective function (Kohonen, Virpioja, and Lagus 2010). It has been shown that the weighting can compensate for a mismatch between the model and the data generating process (Cozman et al. 2003; Cozman and Cohen 2006; Fox-Roberts and Rosten 2014). Therefore, it would appear to be advantageous to study weighting schemes in combination with all the discussed generative models.

5.2 On Potential Novel Approaches

Based on the literature survey presented in Section 3.3.2, one can observe that there exists substantial work on generative lexicon-based approaches and methods based on discriminative boundary detection. In contrast, there exists little to no research on models utilizing lexicons and discriminative learning or generative boundary-detection approaches. In addition, as mentioned in Section 3.3.2, so far there has been little work discussing a combination of lexicon-based and boundary detection approaches. It could be fruitful to explore these modeling aspects further in the future.

6. Conclusions

We presented a comparative study on data-driven morphological segmentation in a minimally supervised learning setting. In this setting the segmentation models are estimated based on a small amount of manually annotated word forms and a large set of unannotated word forms. In addition to providing a literature survey on published methods, we presented an in-depth empirical comparison on three diverse model families. The purpose of this work is to extend the existing literature with a summarizing study on the published methodology as a whole.

Based on the literature survey, we concluded that the existing methodology contains substantial work on generative lexicon-based approaches and methods based on discriminative boundary detection. As for which approach has been more successful, both the previous work and the empirical evaluation presented here strongly imply that the current state of the art is yielded by the discriminative boundary detection methodology. In general, our analysis suggested that the models based on generative lexicon learning are inefficient at utilizing growing amounts of available data. Meanwhile, the studied discriminative boundary detection method based on the CRF framework was successful in gaining consistent reduction in all error types, given increasing amounts of data. Lastly, there exists little to no research on models utilizing lexicons and discriminative learning or generative boundary-detection approaches. Studying these directions could be of interest in future work.

Acknowledgments

This work was financially supported by Langnet (Finnish doctoral programme in language studies), the Academy of Finland under the Finnish Centre of Excellence Program 2012–2017 (grant no. 251170) and LASTU Programme (grants no. 256887 and 259934), project *Multimodally grounded language technology* (grant no. 254104), and the Tiger University program of the Estonian Information Technology Foundation for Education.

References

Brent, Michael R. 1999. An efficient, probabilistically sound algorithm for

segmentation and word discovery. *Machine Learning*, 34(1–3):71–105.

Chrupala, Grzegorz, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with Morfette. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*, pages 2362–2367, Marrakech.

Collins, Michael. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, volume 10, pages 1–8, Philadelphia, PA.

Çöltekin, Çağrı. 2010. Improving successor variety for morphological segmentation. *LOT Occasional Series*, 16:13–28.

Cozman, Fabio and Ira Cohen. 2006. Risks of semi-supervised learning: How unlabelled data can degrade performance of generative classifiers. In Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, editors, *Semi-supervised learning*, pages 57–72. MIT press.

Cozman, Fabio Gagliardi, Ira Cohen, and Marcelo Cesar Cirelo. 2003. Semi-supervised learning of mixture models. In *Proceedings of the 20th International Conference on Machine Learning (ICML-2003)*, pages 99–106, Washington, DC.

Creutz, Mathias, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. 2007. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Transactions on Speech and Language Processing*, 5(1):3:1–3:29.

Creutz, Mathias and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the Workshop on Morphological and Phonological Learning of ACL 2002*, pages 21–30, Philadelphia, PA.

Creutz, Mathias and Krista Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. Technical Report A81, Publications in Computer and Information Science, Helsinki University of Technology.

Creutz, Mathias and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1):1–27.

de Gispert, Adrià, Sami Virpioja, Mikko Kurimo, and William Byrne. 2009. Minimum Bayes risk combination of translation hypotheses from alternative morphological decompositions. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2009)*, pages 73–76, Boulder, CO.

Eger, Steffen. 2013. Sequence segmentation by enumeration: An exploration. *The Prague Bulletin of Mathematical Linguistics*, 100:113–131.

Fox-Roberts, Patrick and Edward Rosten. 2014. Unbiased generative semi-supervised learning. *Journal of Machine Learning Research*, 15(1):367–443.

Goldwater, Sharon. 2006. *Nonparametric Bayesian Models of Lexical Acquisition*. Ph.D. thesis, Brown University.

Green, Spence and John DeNero. 2012. A class-based agreement model for generating accurately inflected translations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 146–155, Jeju Island.

Grönroos, Stig-Arne, Sami Virpioja, Peter Smit, and Mikko Kurimo. 2014. Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 1177–1185, Dublin.

Hammarström, Harald and Lars Borin. 2011. Unsupervised learning of morphology. *Computational Linguistics*, 37(2):309–350.

Harris, Zellig S. 1955. From phoneme to morpheme. *Language*, 31(2):190–222.

Hirsimäki, Teemu, Mathias Creutz, Vesa Siivola, Mikko Kurimo, Sami Virpioja, and Janne Pylkkönen. 2006. Unlimited vocabulary speech recognition with morph language models applied to Finnish. *Computer Speech and Language*, 20(4):515–541.

Jiao, Feng, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL 2006)*, pages 209–216, Sidney.

Johnson, Howard and Joel Martin. 2003. Unsupervised learning of morphology for English and Inuktitut. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL HLT 2003)*, pages 43–45, Edmonton.

Johnson, Mark. 2008. Unsupervised word segmentation for Sesotho using adaptor grammars. In *Proceedings of the 10th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology (SIGMORPHON 2008)*, pages 20–27, Columbus, OH.

Johnson, Mark and Katherine Demuth. 2010. Unsupervised phonemic Chinese word segmentation using Adaptor Grammars. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 528–536, Beijing.

Johnson, Mark and Sharon Goldwater. 2009. Improving nonparameteric Bayesian inference: Experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2009)*, pages 317–325, Boulder, CO.

Johnson, Mark, Thomas L. Griffiths, and Sharon Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *Advances in Neural Information Processing Systems*, pages 641–648, Vancouver.

Johnson, Mark, Thomas L. Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2007)*, pages 139–146, Rochester, NY.

Kaalep, Heiki-Jaan, Kadri Muischnek, Kristel Uiboaed, and Kaarel Veskis. 2010. The Estonian reference corpus: Its composition and morphology-aware user interface. In *Proceedings of the 2010 Conference on Human Language Technologies – The Baltic Perspective: Proceedings of the Fourth International Conference Baltic (HLT 2010)*, pages 143–146, Ŗiga.

Kılıç, Özkan and Cem Bozşahin. 2012. Semi-supervised morpheme segmentation without morphological analysis. In *Proceedings of the LREC 2012 Workshop on Language Resources and Technologies for Turkic Languages*, pages 52–56, Istanbul.

Kohonen, Oskar, Sami Virpioja, and Krista Lagus. 2010. Semi-supervised learning of concatenative morphology. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology (SIGMORPHON 2010)*, pages 78–86, Uppsala.

Kurimo, Mikko, Sami Virpioja, and Ville Turunen. 2010. Overview and results of Morpho Challenge 2010. In *Proceedings of the Morpho Challenge 2010 Workshop*, pages 7–24, Espoo.

Kurimo, Mikko, Sami Virpioja, Ville Turunen, Graeme W. Blackwood, and William Byrne. 2009. Overview and results of Morpho Challenge 2009. In *Working Notes for the CLEF 2009 Workshop*, pages 578–597, Corfu.

Lafferty, John, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, pages 282–289, Williamstown, MA.

Lee, Yoong Keok, Aria Haghighi, and Regina Barzilay. 2011. Modeling syntactic context improves morphological segmentation. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL 2011)*, pages 1–9, Portland, OR.

Lignos, Constantine. 2010. Learning from unseen data. In *Proceedings of the Morpho Challenge 2010 Workshop*, pages 35–38, Helsinki.

Luong, Minh-Thang, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the 17th Conference on Computational Natural Language Learning (CoNLL 2013)*, pages 29–37, Sofia.

Mann, G. and A. McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proceedings of the 46th Annual Meeting of Association for Computational Linguistics: Human Language Technologies (ACL HLT 2008)*, pages 870–878, Columbus, OH.

Monson, Christian, Kristy Hollingshead, and Brian Roark. 2010. Simulating morphological analyzers with stochastic taggers for confidence estimation. In *Multilingual Information Access Evaluation*

I - Text Retrieval Experiments, volume 6241 of *Lecture Notes in Computer Science*.
Narasimhan, Karthik, Damianos Karakos, Richard Schwartz, Stavros Tsakalidis, and Regina Barzilay. 2014. Morphological segmentation for keyword spotting. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 880–885, Doha.
Neuvel, Sylvain and Sean A. Fulop. 2002. Unsupervised learning of morphology without morphemes. In *Proceedings of the 6th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON 2002)*, pages 31–40, Philadelphia, PA.
Nigam, Kamal, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2–3):103–134.
Pirinen, Tommi. 2008. Automatic finite state morphological analysis of Finnish language using open source resources [in Finnish]. Master's thesis, University of Helsinki.
Poon, Hoifung, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2009)*, pages 209–217, Boulder, CO.
Qiu, Siyu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 141–150, Dublin.
Rissanen, Jorma. 1989. *Stochastic Complexity in Statistical Inquiry*, volume 15. World Scientific Series in Computer Science, Singapore.
Ruokolainen, Teemu, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *Proceedings of the 17th Conference on Computational Natural Language Learning (CoNLL 2013)*, pages 29–37, Sofia.
Ruokolainen, Teemu, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2014. Painless semi-supervised morphological segmentation using conditional random fields. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*, pages 84–89. Gothenburg.
Schone, Patrick and Daniel Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies (NAACL 2001)*, pages 1–9, Pittsburgh, PA.
Sirts, Kairit and Sharon Goldwater. 2013. Minimally-supervised morphological segmentation using adaptor grammars. *Transactions of the Association for Computational Linguistics*, 1(May):255–266.
Smit, Peter, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*, pages 21–24, Gothenburg.
Spiegler, Sebastian and Peter A. Flach. 2010. Enhanced word decomposition by calibrating the decision threshold of probabilistic models and using a model ensemble. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, pages 375–383, Uppsala.
Stallard, David, Jacob Devlin, Michael Kayser, Yoong Keok Lee, and Regina Barzilay. 2012. Unsupervised morphology rivals supervised morphology for Arabic MT. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 322–327, Jeju Island.
Sun, Weiwei and Jia Xu. 2011. Enhancing Chinese word segmentation using unlabeled data. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pages 970–979, Edinburgh.
Turian, Joseph, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, pages 384–394, Uppsala.
Turunen, Ville and Mikko Kurimo. 2011. Speech retrieval from unsegmented Finnish audio using statistical morpheme-like units for segmentation, recognition,

and retrieval. *ACM Transactions on Speech and Language Processing*, 8(1):1:1–1:25.
Virpioja, Sami, Oskar Kohonen, and Krista Lagus. 2010. Unsupervised morpheme analysis with Allomorfessor. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments: 10th Workshop of the Cross-Language Evaluation Forum (CLEF 2009)*, pages 609–616, Corfu.
Virpioja, Sami, Oskar Kohonen, and Krista Lagus. 2011. Evaluating the effect of word frequencies in a probabilistic generative model of morphology. In *Proceedings of the 18th Nordic Conference of Computational Linguistics (NODALIDA 2011)*, pages 230–237, Riga.
Virpioja, Sami, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. 2013. Morfessor 2.0: Python implementation and extensions for Morfessor Baseline. Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, Department of Signal Processing and Acoustics, Aalto University, Helsinki, Finland.
Virpioja, Sami, Ville Turunen, Sebastian Spiegler, Oskar Kohonen, and Mikko Kurimo. 2011. Empirical comparison of evaluation methods for unsupervised learning of morphology. *Traitement Automatique des Langues*, 52(2):45–90.
Wang, Yang, Gholamreza Haffari, Shaojun Wang, and Greg Mori. 2009. A rate distortion approach for semi-supervised conditional random fields. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2008–2016, Vancouver.
Wang, Yiou, Yoshimasa Tsuruoka Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving Chinese word segmentation and POS tagging with semi-supervised methods using large auto-analyzed data. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 309–317, Chiang Mai.
Yarowsky, David and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Meeting of the Association for Computational Linguistics (ACL 2000)*, pages 207–216, Hong Kong.
Zhu, Xiaojin and Andrew B. Goldberg. 2009. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130.

Publication III

Stig-Arne Grönroos, Katri Hiovain, Peter Smit, Ilona Rauhala, Kristiina Jokinen, Mikko Kurimo, and Sami Virpioja. Low-resource active learning of morphological segmentation. *Northern European Journal of Language Technology*, Vol. 4, Article 4, pages 47–72, Mar 2016.

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

Low-Resource Active Learning of Morphological Segmentation

Stig-Arne Grönroos[1]
stig-arne.gronroos@aalto.fi

Katri Hiovain[2]
katri.hiovain@helsinki.fi

Peter Smit[1]
peter.smit@aalto.fi

Ilona Rauhala[2]
ilona.rauhala@helsinki.fi

Kristiina Jokinen[2]
kristiina.jokinen@helsinki.fi

Mikko Kurimo[1]
mikko.kurimo@aalto.fi

Sami Virpioja[3]
sami.virpioja@aalto.fi

[1]Department of Signal Processing and Acoustics, Aalto University, Finland
[2]Institute of Behavioural Sciences, University of Helsinki, Finland
[3]Department of Computer Science, Aalto University, Finland

October 10, 2016

Abstract

Many Uralic languages have a rich morphological structure, but lack morphological analysis tools needed for efficient language processing. While creating a high-quality morphological analyzer requires a significant amount of expert labor, data-driven approaches may provide sufficient quality for many applications. We study how to create a statistical model for morphological segmentation with a large unannotated corpus and a small amount of annotated word forms selected using an active learning approach. We apply the procedure to two Finno-Ugric languages: Finnish and North Sámi. The semi-supervised Morfessor FlatCat method is used for statistical learning. For Finnish, we set up a simulated scenario to test various active learning query strategies. The best performance is provided by a coverage-based strategy on word initial and final substrings. For North Sámi we collect a set of human-annotated data. With 300 words annotated with our active learning setup, we see a relative improvement in morph boundary F_1-score of 19% compared to unsupervised learning and 7.8% compared to random selection.

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

1 Introduction

In morphologically rich languages, such as the Uralic languages, the number of observed word forms grows rapidly with increasing corpus size. For instance, in Finnish, nouns can have over 2000 different word forms due to case inflection and various clitics, while verbs can have about 12 000 different forms due to person, number, tempus, and modus inflection and especially to the abundance of infinitival and participle forms, the latter of which are inflected like nouns (Karlsson, 1982). Naturally not all valid combinations of suffixes are common in usage, but they are nevertheless not only theoretical possibilities but part of the living language.

This vocabulary growth can be problematic for natural language processing (NLP) applications, because it causes sparsity in the calculated statistics. Compared e.g. with English which has a small number of inflectional forms, Finnish does not easily lend itself to word form n-gram probability to be used as the basis of NLP tasks since not all possible word forms, not to mention their combinations, occur even in large corpora. Thus it is essential to model such languages on a sub-word level, using for example morphological analysis that allows word forms to be analyzed into parts of two types: the lexical meaning carrying part(s) and the various morphemes which carry grammatical information.

Despite the improvement of development tools and the increase of computational resources since the introduction of finite-state transducer (FST) based morphological analyzers in the 1980s (Koskenniemi, 1983), the bottleneck for the traditional method of building such analyzers is still the large amounts of manual labor and skill that are required (Koskenniemi, 2008). The strength of such analyzers is the potential to produce output of high quality and detailed morphological tags.

Morphological surface segmentation is a relaxed variant of morphological analysis, in which the surface form of a word is divided into segments that correspond to morphemes. The segments, called *morphs*, are not mapped onto underlying abstract morphemes as in FST-based analyzers, but concatenating the sequence of morphs results directly in the observed word form. Allomorphic variation is left unresolved.

Although unsupervised learning of morphological segmenters does not reach the detail and accuracy of hand-built analyzers, it has proven useful for many NLP applications, including speech recognition (Creutz et al., 2007), information retrieval (Kurimo et al., 2010), and machine translation (Virpioja et al., 2007; Fishel and Kirik, 2010; Grönroos et al., 2015b). Unsupervised methods are especially valuable for low-resource languages, as they do not require any expensive resources produced by human experts.

While hand built morphological analyzers and large annotated corpora may be unavailable due to the expense, a small amount of linguistic expertise is easier to obtain. Given word forms embedded in sentence contexts, a well-informed native speaker of a language can mark the prefixes, stems and suffixes of the words in question. A brief collection effort of this type will result in a very small set of annotated words.

A small amount of annotated data of this type can be used to augment a large amount of unannotated data by using semi-supervised methods, which are able to learn from such mixed data. As little as one hundred manually segmented words have been shown to provide significant improvements to the quality of the output when compared to a linguistic gold standard (Kohonen et al., 2010). Adding more annotated data improves the results, with rapid improvement at least up to one thousand words. Ruokolainen et al.

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

(2016) provide an empirical comparison of semi-supervised methods for morphological segmentation.

When gathering annotated training samples for a specific model, *active learning* may provide better results than selecting the samples randomly. A common objective is to reach adequate performance with a shorter annotator effort. In active learning, the annotated words are chosen according to some strategy, making use of information from the available data set, previously selected words, and models trained in previous iterations.

In this work, we use active learning for morphological segmentation of Finnish and North Sámi. This work extends the preliminary results in our previous work (Grönroos et al., 2015a). We extend the work by including experiments in a second language: Finnish. We explore several query strategies for selecting the words to annotate. The comparison to random selection is more rigorously performed.

2 Related work on North Sámi

There has been research effort into FST-based morphology for Sámi languages (Trosterud and Uibo, 2005; Lindén et al., 2009; Tyers et al., 2009). In particular, the Giellatekno research lab[1] provides rule-based morphological analyzers both for individual word forms and running text, in addition to miscellaneous other resources such as wordlists and translation tools. The morphological analyzer gives the morphological properties of a word in the form of tags. For example, given the word ***vaddjojuvvon*** ("cut", PASSIVE), the analyzer produces the following output:[2]

(1)
	vaddjojuvvon	vadjat+V+TV+Der/PassL+V+IV+Ind+Prs+Sg1
	vaddjojuvvon	vadjat+V+TV+Der/PassL+V+IV+Ind+Prt+ConNeg
	vaddjojuvvon	vadjat+V+TV+Der/PassL+V+IV+PrfPrc

Speech technology tools for North Sámi have been explored in the DigiSami project[3] (Jokinen, 2014), which is one of the projects in the Academy of Finland research framework aimed to increase and support digital viability of less-resourced Finno-Ugric languages with the help of speech and language technology. DigiSami focuses especially on North Sámi, and sets to collect data, provide tools, and develop technology to enable North Sámi speech-based applications to be developed (Jokinen and Wilcock, 2014a). Moreover, the project aims to encourage community effort for online content creation, and for this, Wikipedia-based applications are supported, such as WikiTalk (Jokinen and Wilcock, 2014b; Wilcock et al., 2016). This is a robot-application which allows the user to interact with a robot concerning information in the Wikipedia articles.

For speech recognition, a method for statistical segmentation may be preferred over rule-based morphological analyzers. A rule-based analyzer is limited in the vocabulary it recognizes, and non-standard spellings might not be analyzed at all. In addition, the tag set produced by the analyzer may be too rich. For instance, a morphological segmentation of the above example word, ***vaddj + ojuvvo + n***, consists of only 3 morphs, while the Giellatekno analyzer gives a lemma and 6 to 8 tags. Such abstract tags produced by a

[1] `http://giellatekno.uit.no/`
[2] For tag definitions, see `http://giellatekno.uit.no/doc/lang/sme/docu-sme-grammartags.html`
[3] `http://www.helsinki.fi/digisami/`

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

morphological analyzer are not directly applicable in speech recognition, which requires lexical units that can be concatenated into the surface form of the words (Hirsimäki et al., 2006). The work described in this paper directly supports development of the tools that can be used to develop speech technology for North Sámi or other less-resourced languages.

3 On North Sámi and Finnish Morphology

North Sámi (davvisámegiella) belongs to the Finno-Ugric languages and is related to Finnish and other Baltic-Finnic languages. It is one of the nine Sámi languages spoken in the northern parts of Norway, Sweden, Finland and Russia. North Sámi is the biggest of the Sámi languages, with around 30 000 speakers. As the Sámi language speakers do not necessarily understand each other, North Sámi functions as a lingua franca among the Sámi speakers. It is also widely used in newspapers and text books, and there are Sámi language TV and radio broadcasts.

Linguistically, North Sámi is characterized as an inflected language, with cases, numbers, persons, tense and mood. The inflectional system has seven categories: the nouns have four inflection categories (stems with a vowel or a consonant, the so-called contracting is-nouns, and alternating u-nouns), and the verbs have three conjugation categories (gradation, three syllabic, and two syllabic verbs). The only monosyllabic verbs are "leat" (to be) and the negation verb.[4] The verbs and pronouns have specific dual forms besides singular and plural forms, i.e. "we the two of us" and "we more than two".

North Sámi features a complicated although regular morphophonological variation. For instance, the inflected forms follow weak and strong grades which concern almost all consonants. North Sámi is also a fusional language and a single morph can stand for more than one morphological category. In a similar way as in Estonian, loss of certain suffixes has resulted in complicated morphophonological alternations or gradation patterns in the stem. This is especially true of the genitive-accusative form, e.g. *girji* ("book", SgNom) vs. *girjji* ("book", SgGen-Acc).

Adjectives typically have two forms: predicative (*duojár lea* ***čeahppi*** "the craftsman is skillful") and attributive (***čeahpes*** *duojár* "a skillful craftsman") (Sammallahti, 1998). Furthermore, for many adjectives the attributive form can take two alternative forms. For example *seavdnjat* ("dark") has the two attributive variants *sevdnjes* and *seavdnjadis*.

North Sámi has productive compound formation, and compounds are written together without an intermediary space. For example *nállošalbmái* ("into the eye of the needle"), could be segmented as *nállo + šalbmá + i*. North Sámi makes extensive use of derivation, both in verbs and in nouns. For example the adjective *muoŧtái* ("with many aunts") is derived in a regular manner from *muoŧta* ("aunt").

In order to show applicability of the proposed method to another language, we include experiments using Finnish. The choice of Finnish as the second language is motivated by its morphological similarity to North Sámi, making it reasonable to use the results of the Finnish experiment in designing the North Sámi experiment. In addition, we can take advantage of the wide availability of data and tools for Finnish. The Morpho

[4]Like Finno-Ugrian languages in general, also North Sámi forms negation by a particular negation verb which is inflected in person.

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

Challenge data (Kurimo et al., 2007, 2010) processed by a morphological analyzer enable the experiment with a simulated annotator.

North Sámi and Finnish morphology share many similarities. Nouns are inflected by case and can have possessive suffixes attached, while verbs inflect by person, number, tempus and modus. Morphemes are typically ordered according to the same structure, such as in

(2) *kisso +i +lla +nne +kin*
stem PL. ADE. POSS. clitic
bussá +in +eattet +ge
also on your cats

There are also syntactic similarities, such as forming negations using a negation verb. Both languages have gradation of stems. There is even a large number of words with a shared origin, both through the shared origin of the languages and through loaning of words from Finnish to Sámi.

There are also some dissimilarities between the languages, including the dual form for pronouns and verbs in North Sámi, and the number of cases (6 in North Sámi, 15 in Finnish). Adjectives in Finnish do not have a separate attributive form.

Moreover, the morphophonology of the languages differs. North Sámi has neither vowel harmony nor final consonant gemination. North Sámi has 30 consonants, which is more than the 17 in Finnish, but less vowels (7 in North Sámi, 8 in Finnish) (Aikio, 2005; VISK, 2004). In North Sámi, gradation applies to almost all consonants, and thus there is more morphophonological alternation than in Finnish.

4 Annotation of North Sámi Segmentation

Most North Sámi words have an unambiguous segmentation agreeing both with intuition and with established linguistic interpretation. These words contain only easily separated suffixes: markers for case and person, and derivational endings. However, some words require the annotator to make choices on where to place the boundary. In this section, we will describe the challenges faced during annotation, and the decisions made in response.

As a general principle, we aimed to maximize the consistency of the annotations. For established linguistic interpretation we referred to the work by Aikio (2005); Álgutietokanta (2006); Nickel and Sammallahti (2011); Sammallahti (1998).

Inflectional morphology is typically more straightforward to analyze than derivational morphology. The optimal granularity on which to analyze derivations depends on the needs of the application. It appears that the Giellatekno analyzer does not return verbs derived from nouns to the originating noun, even though it does do so for verbs derived from other verbs. We have segmented both derivational and inflectional morphology, without marking the distinction in the segmentation. We deviate from the granularity preferred by Giellatekno by also segmenting derivational suffixes that convert nouns into verbs, if the boundary is distinct.

An exception was made in the case of certain lexicalized stems. These stems appear to end with a derivational suffix, but removal of the suffix does not yield a morpheme at all, or results in a morpheme with very weak semantic relation to the lexicalized stem.

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

An example is *ráhkadi + t* ("make, produce"), rather than *ráhka + di + t*, compared to *ráhka + t* ("crack").

A related challenge was posed by certain lexicalized adverbial forms. These words appear to contain suffixes that could have been segmented, but these suffixes do not have their conventional function in the word. For example, the segmentations *davá + s* ("to the north") and *davvi + n* ("in the north") would appear to contain the singular locative and essive case marker, respectively, but would not have their conventional meanings. A decision was made to leave these forms unsegmented.

To remain consistent, it was rather important that the annotator(s) recognized the declensions of the words. This is because North Sámi has several declensions both in nouns and verbs, and the segmentations often vary depending on them when following grammatical interpretation.

A further challenge was posed by the extensive stem alternation and fusion in Sámi. To maximize consistency, the segmentation boundary was usually placed so that all of the morphophonological alternation remains in the stem. Even though language education classifies verbs into verb types according to the suffix (*-it, -at, -ut, ...*), we have segmented the infinitive marker as *-t*. The preceding vowel is seen as part of the stem, undergoing alternation for phonological and grammatical reasons. A similar decision was needed for the multifunctional derivational ending of verbs, *-d-* or *-di-*. Also, the corresponding literature shows some varying interpretations about these suffixes (Sammallahti, 1998; Nickel and Sammallahti, 2011). For example *boradit* could be segmented both as *bora + di + t* and *bora + d + it*. In this work we have used the former segmentation.

Exceptions include the passive derivational suffix, which is found as variants *-ojuvvo-*, *-juvvo-* and *-uvvo-*, depending on the inflectional category and stem type. The pleonastic derivational ending for actor occurs in the forms *-jeaddji-* and *-eaddji-*.

Observe that many of the segmented suffixes, such as *-i*, and *-t*, occur homonymously in different word classes. For example *-t* could act as a marker for nominative plural in nouns or a marker for present time Sg2 person in verbs, and can also have other functions.

5 Semi-supervised Morphological Segmentation

While unsupervised morphological segmentation has recently been an active topic of research (Hammarström and Borin, 2011), semi-supervised morphological segmentation has not received as much attention. Semi-supervised morphological segmentation can be approached in many ways. One approach is to seed the learning with a small amount of linguistic knowledge in addition to the unannotated corpus (Yarowsky and Wicentowski, 2000). Some semi-supervised methods where a part of the training corpus is supplied with correct outputs have also been presented, including generative (Kohonen et al., 2010; Sirts and Goldwater, 2013; Grönroos et al., 2014) and discriminative (Poon et al., 2009; Ruokolainen et al., 2014) methods.

5.1 Morfessor FlatCat

As a method for morphological segmentation of words, we use Morfessor FlatCat (Grönroos et al., 2014). It is the most recent addition to the Morfessor family of methods for learning morphological segmentations primarily from unannotated data.

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

The method is based on a generative probabilistic model which generates the observed word forms by concatenating morphs. The model parameters $\boldsymbol{\theta}$ define a *morph lexicon*. The morph m_i is considered to be stored in the morph lexicon, if it has a non-zero probability $P(m_i \mid \boldsymbol{\theta})$ given the parameters.

Morfessor utilizes a prior distribution $P(\boldsymbol{\theta})$ over morph lexicons, derived from the Minimum Description Length principle (Rissanen, 1989). The prior favors lexicons that contain fewer, shorter morphs. The purpose is to find a balance between, on one hand, the size of the lexicon, and, on the other hand, the size of the corpus $\boldsymbol{D}$ when encoded using the lexicon $\boldsymbol{\theta}$. This balance can be expressed as finding the following Maximum a Posteriori (MAP) estimate:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta} \mid \boldsymbol{D}) = \arg\min_{\boldsymbol{\theta}} \big(-\log P(\boldsymbol{\theta}) - \log P(\boldsymbol{D} \mid \boldsymbol{\theta}) \big). \tag{3}$$

In order to use the annotations produced in the active learning for training Morfessor, we employ the semi-supervised training approach by Kohonen et al. (2010). This involves replacing the MAP estimate (3) with the optimization

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \big(-\log P(\boldsymbol{\theta}) - \alpha \log P(\boldsymbol{D} \mid \boldsymbol{\theta}) - \beta \log P(\boldsymbol{A} \mid \boldsymbol{\theta}) \big), \tag{4}$$

where $\boldsymbol{A}$ is the annotated training corpus, and α and β are the weights for the likelihood of the unannotated corpus and annotated corpus, respectively. Both the hyper-parameters α and β affect the overall amount of segmentation predicted by the model. The β hyper-parameter also affects the relative importance of using the morphs present in the annotated corpus, compared to forming a segmentation from other morphs in the lexicon.

Morfessor FlatCat uses a flat lexicon, in contrast to the hierarchical lexicon in the Categories-MAP (Creutz and Lagus, 2005) (Cat-MAP) variant of Morfessor. In a hierarchical lexicon, morphs can be built using other morphs already in the lexicon, while in a flat lexicon each morph is represented directly as a string of letters. Each letter requires a certain number of bits to encode, making longer morphs more expensive to add to the lexicon.

A hierarchical lexicon has some benefits in the treatment of frequent strings that are not morphs, but it also presents challenges in model training. When using a flat lexicon, all morph references point from the corpus to the lexicon, making ML estimation of HMM parameters straightforward, and allowing the factorization required for the weighting of the cost function components seen in Equation 4. When using a hierarchical lexicon, also the references *within* the lexicon must be taken into account, making this approach to semi-supervised learning inapplicable.

Moreover, for a flat lexicon, the cost function divides into two parts that have opposing optima: the cost of the data (the likelihood) is optimal when there is minimal splitting and the lexicon consists of the words in the training data, whereas the cost of the model (the prior) is optimal when the lexicon is minimal and consists only of the letters. In consequence, the balance of precision and recall of the segmentation boundaries can be directly controlled by weighting the data likelihood using the hyper-parameters. Tuning these hyper-parameters is a very simple form of supervision, but it has drastic effects on the segmentation results (Kohonen et al., 2010). A direct control of the balance may also be useful for some applications: Grönroos et al. (2015b) used this method to tune segmentation for machine translation.

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

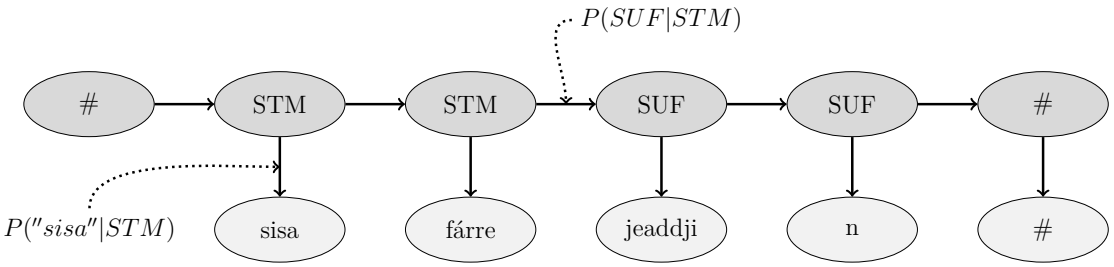


Figure 1: A graph representation of the Hidden Markov model morphotactics, applied to the example word *sisafárrejeaddjin*. The word boundary symbol is marked #. One transition and one emission probability are indicated.

Figure 1 illustrates the hidden Markov model (HMM) used for modeling word formation. The HMM has morph categories as hidden states and morphs as observations. Each morph token is categorized as prefix (PRE), stem (STM), or suffix (SUF). Internally to the algorithm, a non-morph (NON) category is used, intended to model frequent substrings that are not morphs but fragments of a morph. HMM morphotactics were previously used in the Categories-ML (Creutz and Lagus, 2004) and Cat-MAP variants of Morfessor, but Morfessor FlatCat is the first method to combine the approach with semi-supervised training.

In order to calculate the emission probability of a morph conditioned on the morph category, $P(m_i \,|\, c_i)$, the prior of Morfessor FlatCat includes encoding of the right and left perplexity of the morph. The perplexity measures describe the predictability of the contexts in which the morph occurs. Morphs with unpredictable right or left contexts are more likely to be prefixes or suffixes, respectively. Longer morphs are more likely to be stems. The perplexities and length in characters are turned into probabilities, by applying a sigmoidal soft thresholding followed by normalization.

The benefit of the HMM morphotactics is increased context-sensitivity, which improves the precision of the segmentation. For example, in English, the model can prevent splitting a single *s*, a common suffix, from the beginning of a word, e.g. in **s + wing*. Modeling of morphotactics also improves the segmentation of compound words, by allowing the overall level of segmentation to be increased without increasing over-segmentation of stems. The presence of morph categories in the output makes it simple to use the method as a stemmer by removing affixes and retaining only stems. The main benefits of semi-supervised learning are in the modeling of suffixation. As the class of suffixes is closed and has high frequency, a good coverage can be achieved with a relatively small set of annotations, compared to the open morph classes such as compound parts. (Grönroos et al., 2014)

The model parameters $\boldsymbol{\theta}$ are optimized utilizing a greedy local search. In each step, a particular subset of the boundaries is reanalyzed and the model parameters updated.

Morfessor FlatCat is initialized using the segmentation from the 2.0 version (Virpioja et al., 2013) of Morfessor Baseline (Creutz and Lagus, 2002, 2007). It employs a morph lexicon $P(m \,|\, \boldsymbol{\theta})$ that is simply a categorical distribution over morphs m, in other words a unigram model.

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

6 Active Learning

Data annotation is often performed for the specific goal of improving the performance of a particular system on a task. This gives the opportunity to carefully select the data that will be annotated, in order to maximize the effect and minimize the cost of annotations. This annotation process with systematic (active) data selection is called *active learning*. Many algorithms and methods exist for active learning, but they are not all equally suitable in every situation.

Active learning methods can be divided into three frameworks: pool-based active learning, (membership) query synthesis and stream-based selective sampling (Settles, 2009).

In *pool-based active learning* (Lewis and Gale, 1994), the system has access to a pool of unlabeled data $\mathcal{A}$ and can request from the annotator true labels for a certain number of samples in the pool. Pool-based active learning can be performed either on-line by selecting one sample in each iteration, or as a batch algorithm by selecting a list of samples at once, before updating the information available to the learner. Pool-based active learning has been successfully applied in NLP (McCallumzy and Nigamy, 1998).

Pool-based active learning can be contrasted with *query synthesis*, in which the learner generates samples to annotate de novo, instead of selecting from a pool of candidates. These methods are difficult to apply to morphological segmentation, due to the challenge of generating valid surface forms.

The third category, *stream-based selective sampling*, is similar to pool-based active learning in that there is a pool of potential samples. In this framework, the samples come in one by one, and the learner has to decide in an on-line fashion whether to query an annotation for that sample or not.

In this work we apply pool-based active learning. Therefore we define the active learning procedure as follows:

In each iteration of active learning, a query strategy is applied for selecting the next samples to elicit and add to the annotated data. The query strategy has access to four sources of information that can be used for guiding the decision at time t:

1. the training pool $\mathcal{A}$,
2. the set of unannotated data $\boldsymbol{D}$,
3. the current set of annotated data $\boldsymbol{A}^{<1\ldots t>}$,
4. and the current best model trained with all training samples collected up to that point $\mathcal{M}^{<t>}$.

$$A^{<t+1>} = \textsc{Strategy}(\mathcal{A}, \boldsymbol{D}, \boldsymbol{A}^{<1\ldots t>}, \mathcal{M}^{<t>}) \tag{5}$$

In this work we make a distinction between the training pool $\mathcal{A}$, and the entire unannotated data $\boldsymbol{D}$, even though they are often chosen to be the same set.

More general reviews of active learning have been written by Settles (2009) and Guyon et al. (2011).

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

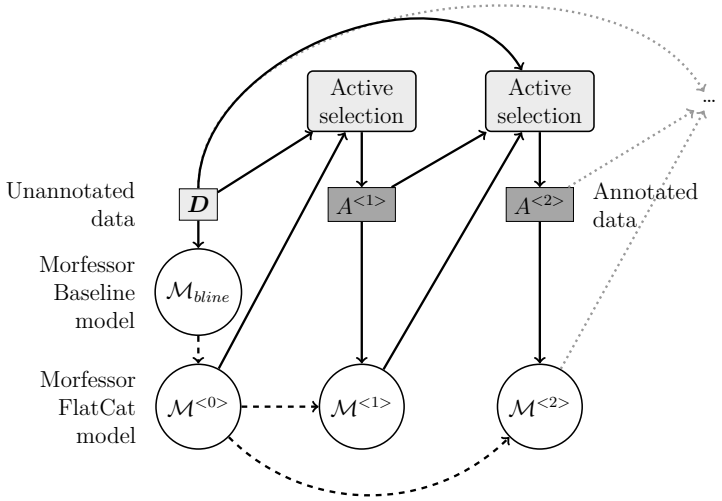


Figure 2: The two first iterations of the active learning procedure applied in this work. Dashed lines indicate the initialization of models. The dotted lines indicate that the procedure can be repeated for additional iterations.

6.1 Active Learning Applied to Morphological Segmentation

Active learning methods have been applied for constructing FST-based analyzers by eliciting new rules from a user with linguistic expertise (Oflazer et al., 2001; Bosch et al., 2008). These development efforts are fast for rule-based systems, but still require months of work.

In the case of morphological segmentation, we try to assess the value of adding the gold standard segmentation of new words into the annotated data set. The methods have access to a list of the n current best segmentations $Z^{<t>}_{i,(1)} \dots Z^{<t>}_{i,(n)}$ for each word w_i, together with their likelihoods given the current model.

Figure 2 shows our active learning procedure, which starts from nothing but an unannotated corpus collected for other purposes. An initial model is trained in an unsupervised fashion. The procedure then applies three components iteratively:

1. active selection of new words to annotate using the query strategy,
2. elicitation of annotations for the selected words, and
3. training of the new segmentation model using all available training data.

7 Query Strategies

Active learning requires a specific method for ranking the samples according to their informativeness. Finding the true informativeness of a sample would require looping over all samples in the pool, eliciting an annotation for the sample and training a new model

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

with only that sample added. As the cost of finding the true informativeness would completely negate the benefits, we need a surrogate objective function that is feasible to optimize. This surrogate objective together with the method for optimizing it is called the *query strategy*. The ranking of the training pool according to this query strategy can then be used for selecting data.

Query strategies fall into two broad categories: strategies that primarily use the previously trained model for the task at hand in order to estimate the objective function, and strategies that define the surrogate objective function separately, by directly modeling the properties of the training set.

In the following section we describe a set of query strategies that are applicable to active learning using Morfessor.

7.1 Uncertainty Sampling

Lewis and Gale (1994) introduced *uncertainty* sampling, which is one of the most commonly used methods (Settles, 2009; Guyon et al., 2011). It was used for the NLP tasks of document classification by Lewis and Catlett (1994), and parsing and information extraction by Thompson et al. (1999).

Uncertainty sampling uses the model's estimate of the uncertainty of the decision associated with a particular sample in order to select the additional samples to annotate. The certainty is given by the likelihood of the current best segmentation, compared to all alternative segmentations.

The next word to annotate $A^{<t+1>}$ at time step t is selected from $\mathcal{A}$ based on the uncertainty of the current best segmentation $Z^{<t>}_{i,(1)}$ for each word w_i

$$\begin{aligned} A^{<t+1>} &= \arg\max_{w_i \in \mathcal{A}} \left[1 - P(Z^{<t>}_{i,(1)} \,|\, w_i; \boldsymbol{\theta}^{<t>})\right] \\ &= \arg\min_{w_i \in \mathcal{A}} \frac{P(Z^{<t>}_{i,(1)}, w_i \,|\, \boldsymbol{\theta}^{<t>})}{P(w_i \,|\, \boldsymbol{\theta}^{<t>})}, \end{aligned} \tag{6}$$

where the likelihood of the word with the current best segmentation $P(Z^{<t>}_{i,(1)}, w_i \,|\, \boldsymbol{\theta}^{<t>})$ is given by the Viterbi algorithm (Viterbi, 1967) and the likelihood of the word with any segmentation $P(w_i \,|\, \boldsymbol{\theta}^{<t>})$ is given by the forward algorithm (Baum, 1972).

7.2 Margin Sampling

While uncertainty sampling compares the probability of the current best segmentation to all alternative segmentations, margin sampling (Scheffer et al., 2001) only compares to the second best alternative segmentation. The distance to the runner up is called the *margin*. If the margin is large, the model is certain about the segmentation. Therefore, the word with the smallest margin is selected.

$$\begin{aligned} A^{<t+1>} &= \arg\min_{w_i \in \mathcal{A}} \left[P(Z^{<t>}_{i,(1)} \,|\, w_i; \boldsymbol{\theta}^{<t>}) - P(Z^{<t>}_{i,(2)} \,|\, w_i; \boldsymbol{\theta}^{<t>})\right] \\ &= \arg\min_{w_i \in \mathcal{A}} \frac{P(Z^{<t>}_{i,(1)}, w_i \,|\, \boldsymbol{\theta}^{<t>}) - P(Z^{<t>}_{i,(2)}, w_i \,|\, \boldsymbol{\theta}^{<t>})}{P(w_i \,|\, \boldsymbol{\theta}^{<t>})} \end{aligned} \tag{7}$$

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

7.3 Query-by-Committee by Bracketing the Corpus Weight

In the *query-by-committee* (QBC) algorithm (Seung et al., 1992; Freund et al., 1997), a committee of predictors independently give their prediction for each sample. The samples that cause most disagreement among the committee members are considered most informative to annotate.

In this experiment the committee consists of two Morfessor FlatCat models, trained with the corpus coding weight hyper-parameter α set to values 10% above and below the optimal value. The reasoning is that the uncertainty about segmentations that are sensitive to a small shift in α is steering the hyper-parameter optimization. Annotating some of these words may allow the global benefits of a slightly different α without introducing errors in words containing the particular morphs in these annotations.

The algorithm filters the words in the training pool, leaving only the words that were segmented differently by the two models in the committee.

$$\mathcal{A}' = \{w_i \in \mathcal{A} : \mathcal{M}_1^{<t>}(w_i) \neq \mathcal{M}_2^{<t>}(w_i)\} \tag{8}$$

In order to select a particular word from the set of filtered words, we pick the one with largest sum of likelihoods given by the two models

$$A^{<t+1>} = \underset{w_i \in \mathcal{A}'}{\arg\max} \left[P(Z_{i,(1)}^{<t>} \mid \boldsymbol{\theta}_1^{<t>}) + P(Z_{i,(1)}^{<t>} \mid \boldsymbol{\theta}_2^{<t>}) \right]. \tag{9}$$

This selects a word that has high likelihood under both models, but that the models still disagree on.

7.4 Coverage of Initial/Final Substrings

The *Initial/final substrings* query strategy is inspired by the feature selection method called *coverage* by Druck et al. (2009), which aims to select features that are dissimilar from existing labeled features, increasing the labeled features' coverage of the feature space.

The method aims to select samples representative of the whole data distribution, instead of querying uncertain samples under the current model, which are likely to contain outliers and exceptional cases.

We apply the idea of coverage to selection of samples to annotate, by defining binary features for the words, and then selecting words so that the features present in them maximize coverage. Our active learning selection differs from the feature selection in that only one sample is needed to cover a feature, instead of labeling all samples with that feature.

We define the features to be substrings starting from the left edge (initial) or ending at the right edge (final) of the word. The length of substrings is limited to between 2 and 5 characters. Let $\Omega(w_i)$ be the set of such substring features in word w_i.

When ranking the words, points are awarded for each substring s present in the ranked word, unless that substring already occurs in the previously selected words. This can be written as the maximization

$$A^{<t+1>} = \underset{w_i \in \mathcal{A}}{\arg\max} \sum_{s \in \Omega(w_i)} \mathrm{I}\left(s \notin \Omega(A^{<j>}) \; \forall j \in \{1 \ldots t\}\right) \frac{\#(s)}{N_{|s|}} \tag{10}$$

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

where I is the indicator function, and $\#(s)$ the occurrence count of feature s. Dividing by

$$N_k = \frac{\sum_s \mathrm{I}(|s| = k)\,\#(s)}{\sum_s \mathrm{I}(|s| = k)} \tag{11}$$

normalizes the occurrence counts by the average occurrence count for substrings of the same length.

This query strategy differs from the other compared methods in that it does not use the Morfessor model when selecting words. However, it can be considered an active selection strategy, as it does define a surrogate objective to systematically take into account the available data $\mathcal{A}$ and the previous selections $\boldsymbol{A}^{<1\ldots t>}$. A benefit of this strategy is that the user does not have to interleave elicitation and Morfessor training. A large list of words can be selected in advance.

7.5 Words without Stem

No stem is a query strategy specific to Morfessor FlatCat. It uses the morph category tags in the current best analysis, to filter a smaller set of potential words from the pool.

Only words for which the current analysis does not contain any morph categorized as STM are kept. This finds stems that are not yet included in the lexicon, and therefore have been over-segmented into NON:s. This improves the coverage of the morph lexicon.

The uncertainty measure is used for selecting individual words from the filtered set.

7.6 Consequent Non-morphemes/Suffixes

Consequent NON/SUF is another strategy specific to Morfessor FlatCat. It is similar to the *No stem* strategy, filtering words to only the words with two or more consecutive morphs categorized as NON or SUF. This strategy is designed to improve suffix chains, in addition to finding over-segmented stems.

7.7 Representative Sampling

Xu et al. (2003) introduce *representative sampling* (RS), that selects samples which are dissimilar to each other, in order to give a good coverage of the dataset.

Selecting dissimilar samples is of particular importance when selection and training is done in batches instead of on-line. An on-line algorithm updates the uncertainty after each sample, making it less likely to select redundant words than a batch algorithm.

We apply representative sampling by clustering the 500 top ranked words for the *Uncertainty* and *QBC* strategies. We cluster the words using k-medoids, with k set to 50. Levenshtein distance (Levenshtein, 1966) is used as the string edit distance. The clustering is repeated 10 times, and the clustering with the smallest intra-cluster variation is selected. The final selection consists of the 50 cluster medoid words.

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

8 Evaluation

The word segmentations generated by the model are evaluated by comparison with annotated morph boundaries using *boundary precision*, *boundary recall*, and *boundary* F_1*-score* (see, e.g., Virpioja et al., 2011). The boundary F_1-score equals the harmonic mean of precision (the percentage of correctly assigned boundaries with respect to all assigned boundaries) and recall (the percentage of correctly assigned boundaries with respect to the reference boundaries).

$$\text{Precision} = \frac{\#(\text{correct})}{\#(\text{proposed})}; \qquad \text{Recall} = \frac{\#(\text{correct})}{\#(\text{reference})} \tag{12}$$

Precision and recall are calculated using macro-averages over the words in the evaluation set. In the case that a word has more than one annotated segmentation, we take the one that gives the highest score.

We also report the scores for subsets of words consisting of different morph category patterns found in the evaluation set. These categories are words that should not be segmented (STM), compound words consisting of exactly two stems (STM+STM), a stem followed by a single suffix (STM+SUF) and a stem and exactly two suffixes (STM+SUF+SUF). Only precision is reported for the STM pattern, as recall is not defined for an empty set of true boundaries.

In addition to the annotated data, we can consider the analysis produced by the North Sámi morphological analyzer from Giellatekno as a secondary gold standard. However, comparing a morphological segmentation to a morphological tagging is not trivial. First, tagging provides abundant information not present in the surface forms. Second, even for tags that have an approximately corresponding morph in the word form, the mapping between the tags and morphs is unknown and must be inferred.

Virpioja et al. (2011) describe several methods for the latter problem. We did preliminary tests with the CoMMA-B1 score that is based on the co-occurrence of the morphemes between the word forms. From the Giellatekno analyses, we split the word forms according to the marked compound boundaries and selected a subset of tags related to inflections and derivations. Then we ran CoMMA-B1 using the annotated test set words as predictions and the modified Giellatekno analyses as a gold standard. This provided precision 0.818, recall 0.155, and F_1-score 0.261. While the scores are also affected by the annotation decisions explained in Section 4, especially the low recall demonstrates that evaluating morphological segmentation based on morphological tagging is problematic.

9 Experiment I: Comparison of Query Strategies

For this experiment, we simulate an annotator using 500 000 segmented word types sampled from the Morpho Challenge 2007 (Kurimo et al., 2007) Finnish data set. This set was analyzed using the two-level morphology analyzer FINTWOL by Lingsoft, Inc., after which the analysis was mapped from the morpheme tags to surface forms of morphemes. This mapping is nontrivial due to the abundance of morphological tags with no surface representation, fusional morphemes, and allomorphy. The applied mapping is described by Creutz and Lindén (2004).

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

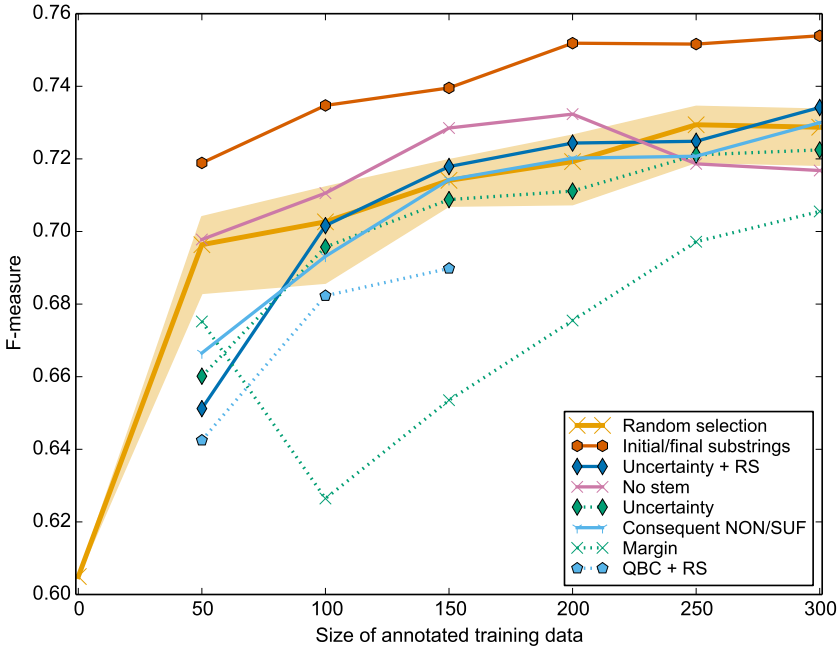


Figure 3: Comparison of different query strategies. The y-axis shows the performance evaluated using F_1-score for the Finnish test set, for models trained using varying amounts of annotated data selected using the query strategy. The thick orange crossed line shows the average of 5 random selections, while the shaded area shows the maximum and minimum.

For hyper-parameter optimization, we used the 835 words in the Morpho Challenge 2010 (Kurimo et al., 2010) development set, and for evaluation the 224 939 words in the corresponding test set.

We simulated an active learning setup using the large annotated data set, by applying the query strategies, and then constructing annotated training sets of the selected words with their annotations. The query strategies did not have access to the annotations of the complete data set.

Regarding the hyper-parameters of Morfessor FlatCat, the corpus likelihood weight α was set by grid search for each selection and iteration individually. In order to considerably decrease the amount of computation, the value for the annotation likelihood weight β was set using a heuristic formula optimized for Finnish:[5]

$$\log \beta = 1.9 + 0.8 \log |\boldsymbol{D}| - 0.6 \log |\boldsymbol{A}|, \tag{13}$$

where $|\boldsymbol{D}|$ and $|\boldsymbol{A}|$ are the numbers of word types in the unannotated and annotated training data sets, respectively. Although it is not guaranteed to be optimal, using the same heuristic value for all query strategies is not expected to favor any particular strategy. The perplexity threshold was set to 75.

[5]The formula is based on work currently being prepared for publication by the present authors.

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

Table 1: Sizes of the unannotated corpora used in Experiment II, and the initial division into subsets.

Corpus	**Word tokens**	**Word types**
Den samiske tekstbanken	17 985 140	691 190
UIT-SME-TTS	42 150	8194
Development set	–	200
Evaluation pool	–	800
Training pool $\mathcal{A}$	–	7194

9.1 Results

Figure 3 shows the F_1-score for different query strategies with increasing amounts of annotations. The random selection baseline is averaged over 5 runs.

The only query strategy that consistently performs better than random selection is *Initial/final substrings*. It appears to plateau after 200 annotated words. Inspection of the selected words reveals an assortment of words with common suffixes and compound modifiers.

The *No stem* strategy initially shows strong performance, but falls below random selection when 250 or more words are annotated.

For the *Uncertainty* strategy, applying the *Representative sampling* improves performance, but it should be noted that when only 50 words have been selected, it performs worse than random selection. These first selected words appear to contain many outliers.

Margin sampling does not perform well when used with Morfessor FlatCat. Some selected words have a small margin due to small differences in the category tagging of morphs, which does not even affect the segmentation. Other words are outlier non-words, with several low-probability segmentation alternatives. Margin sampling would also benefit from applying the representative sampling, as it tends to select many words that are similar to each other.

For the Query-by-Committee (QBC) strategy, only the best results which included representative sampling (RS) are plotted. The method performed worse than random selection, and was discontinued after 3 iterations. The selections of this strategy consisted entirely of compound words, with much redundancy in compound parts despite the representative sampling.

Based on these results, *Initial/final substrings* was selected as the main query strategy for the North Sámi experiment. *Uncertainty+RS* was also included, due to its popularity in the literature, and receiving the second highest score at 300 annotated words.

10 Experiment II: Active Learning for North Sámi

We used two different text corpora in our experiments. The sizes of the corpora are shown in Table 1. The larger *Den samiske tekstbanken* corpus[6] was only used as source

[6]Provided by UiT, The Arctic University of Norway.

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

for a word list, to use as the unannotated training data. It contains texts of six genres: administrative, bible, facta, fiction, laws and news.

The smaller *UIT-SME-TTS* corpus was divided into separate pools from which evaluation and training words were drawn for annotation. The sentences in which the words occur were also extracted for use as contexts. To ensure that the evaluation words are unseen, the words in the evaluation pool were removed from the other subsets.

The use of two corpora enables the release of the annotations with their sentence contexts. Selecting sentences from *Tekstbanken* would have precluded release, as the restrictive license of the *Tekstbanken* corpus does not allow republication. It also demonstrates the effectiveness of the system under the realistic scenario where a large general-domain word list for the language is available for use, even though the corpora themselves are unavailable due to restrictive licensing. A similar scenario would be selection from a specific target domain corpus.

In contrast to our preliminary work (Grönroos et al., 2015a) we used Morfessor FlatCat during the entire experiment. We used Morfessor Baseline only as initialization method for the initial Morfessor FlatCat model. FlatCat models in later iterations were initialized from the unsupervised FlatCat model, as shown in Figure 2.

As prefixes are very rare in North Sámi, and none were seen in the annotations, we disabled the prefix category by setting an extremely high perplexity threshold for prefixes.

In contrast to the Finnish used in Experiment I, we did not have a heuristic formula for β similar to Equation 13 that would be suitable for North Sámi. However, as we had a smaller number of compared methods, we could set all three hyper-parameters (corpus likelihood weight α, annotation likelihood weight β, perplexity threshold for suffixes) by a grid search for each selection and iteration individually.

10.1 Elicitation of Annotations

In this section, we describe the tool used for elicitation during this experiment, and the resulting data set. For discussion on the challenges of annotating North Sámi for morphological segmentation and our responses to them, see Section 4.

There are no efficient on-line training algorithms for Morfessor FlatCat. Thus we used a batch procedure, by selecting a list of 50 new words to annotate with the query strategy being evaluated, and re-trained Morfessor once the whole list had been annotated.

As the *Initial/final substrings* query strategy does not depend on the Morfessor model during active selection, it was possible to evaluate in a single iteration selecting and annotating the full list of 300 words. Subsets were also evaluated, to show the effect of varying the size of annotations.

For the elicitation step, we developed a web-based annotation interface. A javascript app using the jQuery framework was used as a front-end and a RESTful Python wsgi-app built on the bottle framework[7] as a back-end. For words in the training pool, the interface shows the segmentation of the current model as a suggestion to the annotator. Words in the development and evaluation pools are shown unsegmented, in order not to bias the annotator.

The tool gives the option of providing a distinct segmentation for word tokens with the same surface form, depending on the sentence context. Even word forms belonging to

[7] `http://bottlepy.org/`

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

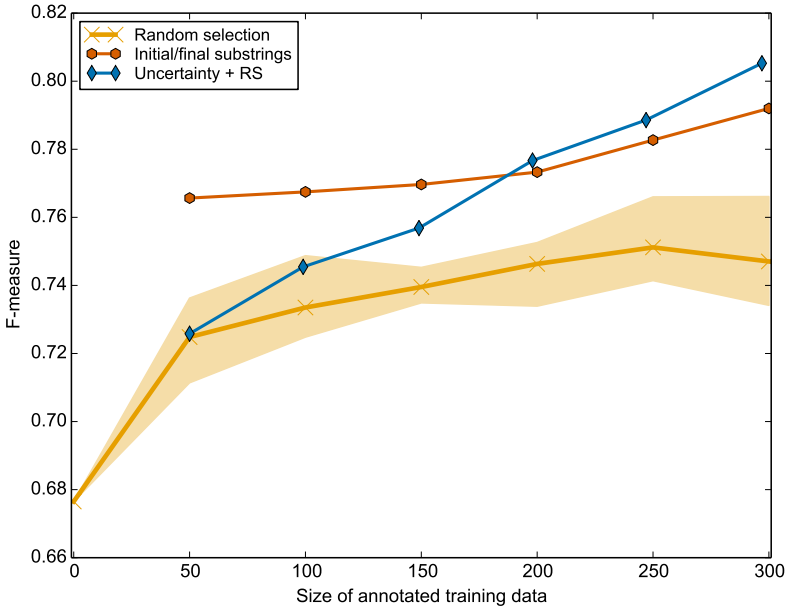


Figure 4: Evaluation using F_1-score for the North Sámi test set, for models trained using varying amounts of annotated data selected using the two selected query strategies. The thick orange crossed line shows the average of 5 random selections, while the shaded area shows the maximum and minimum.

different parts of speech could be homonymous through inflection, and therefore require phrasal context to disambiguate. For example *vearrái* would be unsegmented if it occurs as an adjective ("mean, evil"), but would be segmented *vearrá + i* if it occurs as the illative of the noun *vearri* ("mistake, wrongdoing").

In some rare cases there was no phrasal context provided with the word to be segmented, making it impossible to disambiguate between possible alternative segmentations. This could be caused by isolated words in the corpus, or by mistakes in the automatic tokenization. In these cases, the annotator had to make a judgment call on how to disambiguate the word token.

The annotations were produced by a single Sámi scholar, who is not a native speaker of Sámi. In total 2311 annotated words were collected, divided into 1493 randomly selected word types and 818 actively selected word types. The total time spent by the annotator was 32 hours.[8] A second non-native Sámi speaking linguist independently reannotated 815 of the same words. The principles for segmentation of ambiguous words were discussed prior to the reannotation, but the work itself was independent. Comparing the placement of morph boundaries in the annotations using Cohen's kappa (Cohen, 1960) results in an inter-annotator agreement of 0.82 ("almost perfect agreement").

[8]Includes time spent during the preliminary experiments. Breaks longer than 30 minutes are omitted.

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

Table 2: The model parameters, number of annotated words, and North Sámi test set BPR, for models trained in each iteration of Experiment II. For random selection the averages over all repetitions are shown. Note that the size of the annotated data set may be less than the number of selected words, if non-words were selected.

		Hyper-parameters			**Full test set**		
Model	$\|\boldsymbol{A}\|$	α	β	ppl-thresh	Pre	Rec	F_1
Unsupervised	0	0.4	–	20	0.726	0.633	0.677
Random selection	50	0.48	18000	40	0.725	0.725	0.725
Initial/final substrings	50	0.4	15000	40	**0.733**	**0.802**	**0.766**
Uncertainty + RS	50	0.3	21000	25	0.687	0.769	0.726
Random selection	100	1.02	18000	40	0.746	0.721	0.734
Initial/final substrings	100	1.5	23000	40	**0.765**	**0.769**	**0.767**
Uncertainty + RS	99	0.8	16000	30	0.732	0.760	0.745
Random selection	150	1.30	15000	40	0.754	0.726	0.740
Initial/final substrings	150	1.7	19000	40	**0.774**	**0.766**	**0.770**
Uncertainty + RS	149	1.4	15000	60	0.757	0.757	0.757
Random selection	200	1.32	16000	40	0.750	0.743	0.746
Initial/final substrings	200	1.9	18000	40	0.767	**0.780**	0.773
Uncertainty + RS	198	1.7	14000	70	**0.776**	0.778	**0.777**
Random selection	250	1.56	14000	40	0.760	0.743	0.751
Initial/final substrings	250	1.7	16000	50	0.766	0.800	0.783
Uncertainty + RS	247	1.5	15000	80	**0.768**	**0.811**	**0.789**
Random selection	300	1.68	10000	40	0.763	0.732	0.747
Initial/final substrings	300	1.4	14000	40	0.767	0.819	0.792
Uncertainty + RS	297	1.5	14000	80	**0.772**	**0.842**	**0.805**

10.2 Results

Figure 4 shows the improvement of the F_1-score as more annotations became available. The random selection baseline was averaged over 5 repetitions.

As in Experiment I, the *Initial/final substrings* strategy performs consistently better than random selection. In contrast to that experiment, its performance does not stagnate, but accelerates in the last two iterations.

The results for the *Uncertainty + Representative sampling* strategy differ in several ways from Experiment I. While performance at 50 words is again weak, it is at no iteration worse than random selection. Performance increases rapidly, with Uncertainty + RS surpassing Initial/final substrings when 200 words have been selected.

Table 2 shows the models trained in this experiment. For the full test set, we improve the F_1-score by 18.9% compared to unsupervised learning, with most of the improvement coming from an increase in recall. There is also a small increase in precision. Compared to random selection, the increase in F_1-score is 7.8%.

The values for the hyper-parameters are also shown in Table 2. The optimal value for the corpus likelihood weight α is different for unsupervised and semi-supervised training,

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

Table 3: Boundary precision (Pre), recall (Rec), and F_1-scores for different subsets of the evaluation data.

		STM	**STM+STM**			**STM+SUF**			**STM+SUF+SUF**		
Words in subset		228	55			335			65		
Model	$\|\boldsymbol{A}\|$	Pre	Pre	Rec	F_1	Pre	Rec	F_1	Pre	Rec	F_1
Unsupervised	0	.697	.897	.836	.866	.664	.427	.520	.715	.369	.487
Random selection	50	.648	.869	.848	.859	.696	.587	.637	.712	.433	.538
Initial/final substr.	50	.645	.827	**.909**	.866	.717	.716	.717	.733	.500	.595
Uncertainty + RS	50	.579	.820	.891	.854	.665	.654	.659	.751	.477	.583
Random selection	300	**.705**	**.899**	.807	.851	.739	.608	.667	.711	.477	.571
Initial/final substr.	300	.675	.842	.855	.848	.774	.743	.759	.715	.569	.634
Uncertainty + RS	297	.667	.867	.873	**.870**	**.777**	**.779**	**.778**	**.769**	**.638**	**.698**

with the change happening between 50 and 100 words. The same phenomenon could be seen in Experiment I. Different local optima of α seem to be dominant, depending on the influence of the annotations. Despite the decrease in overall segmentation caused by this increase in α, the semi-supervised models segment ca 15% more than the unsupervised model.

Statistical significance testing was performed using the Wilcoxon signed-rank test ($p < 0.01$). The difference between *Initial/final substrings* and random selection was shown to be statistically significant for all sizes of annotated data. The difference between *Uncertainty + RS* and random selection was only significant with 200 annotated words or more. The difference between the two active selection strategies was only significant at 50 annotated words.

Table 3 shows scores for different categories of words, defined using patterns of morph categories. The selected patterns include all patterns with two morphs or less. For these patterns, precision and recall have a straightforward interpretation. The STM+SUF+SUF pattern was included to shed light on the handling of the boundary between two suffixes. The selected patterns cover 86% of the words in the test set.

When comparing to unsupervised learning, all three forms of semi-supervised learning give better results for suffixation (STM+SUF and STM+SUF+SUF), already with just 50 annotated words. The score for words without internal structure (STM pattern) is only improved when selecting 300 words randomly. For both suffix patterns, active selection is superior to random selection, especially in recall. However, recall for the STM+SUF+SUF remains low for all compared systems. The boundary between two suffixes is the most difficult for Morfessor to place correctly (Ruokolainen et al., 2016).

Random selection gives the best precision for compound words (STM+STM), but has low recall also for this pattern.

After excluding the STM pattern, the best performing method is unambiguous for a particular number of annotations. If only 50 annotated words are used, the best performance for all remaining patterns is given by the *Initial/final substrings* strategy. With 300 annotated words, the best performing strategy is *Uncertainty + RS*.

Initial/final substrings assumes that one sample is enough to cover a feature. In other

Northern European Journal of Language Technology, 2016, Vol. 4, Article 4, pp 47–72
DOI 10.3384/nejlt.2000-1533.1644

words, it assumes that every word beginning or ending with a particular substring will equally well teach the model how to segment other words with the same substring. In practice this assumption does not hold, e.g. *seammaláhkái* ("by the same means") is segmented *seamma + láhkái*, while *govvadahkkái* ("to the picture maker") is segmented *govva + dahkká + i*. Both words end in *kái*, but it is segmented differently. The query strategy is to some extent able to compensate by using longer substrings, and in practice does not seem to make too many detrimental selections.

11 Conclusions

We have applied an active learning approach to modeling morphological segmentation of two Uralic languages: Finnish and North Sámi. The work was accomplished using open-source software.[9] We present the collected language resources for the use of the scientific community.[10]

We performed two experiments. In the first experiment, we compared seven different query strategies using Finnish gold standard segmentations to simulate an annotator. In the second experiment, we applied the active learning system to collect a set of human-annotated data for North Sámi.

In both of the experiments, the *Initial/final substrings* query strategy performed better than random selection regardless of the size of the annotated data set. In the Finnish language experiment, it is clearly the best method.

The performance of the segmentation model was shown to increase rapidly as the amount of human-annotated data was increased. With 300 annotated North Sámi words, collected using the *Uncertainty + Representative sampling* query strategy, F_1-score was improved by 19% (relative) compared to unsupervised learning and 7.8% (relative) compared to random selection. The increase was consistent over several sets of words with different morphological patterns. The largest benefit of the annotations was in the modeling of suffixation.

The results of the two experiments differ with regard to the performance of the *Uncertainty + RS* query strategy. In the last iterations of the North Sámi experiment, it outperforms Initial/final substrings, even though the difference is not statistically significant. The different outcomes may be caused by real differences between the morphology of the languages, or the properties of the data sets. However, the difference could also be an artifact caused by either the procedure of simulating an annotator or the heuristic hyper-parameter values used in Experiment I.

If the proposed method is applied to a new language, the minimum amount of training set words to annotate should be around 100, in addition to the development set needed for the hyper-parameter optimization. The transition of the value of the hyper-parameter α from the local optimum of unsupervised training to the optimum of semi-supervised training has not yet occurred at 50 annotated words. Additionally, with only 50 annotated words, *Uncertainty + RS* does not yet outperform random selection. If a very small number of words (100–200) are collected, we recommend using the *Initial/final substrings* query strategy. As the number of annotations grows larger, active selection is

[9]Morfessor is available at `http://www.cis.hut.fi/projects/morpho/`.
The annotation and active learning tool is available at `https://github.com/Waino/morphsegannot/`.

[10]The data is available at `http://research.spa.aalto.fi/speech/data_release/north_saami_active_learning/`.

still preferable over random selection, but the choice of specific query strategy may be less important.

The Initial/final substrings query strategy does not apply the current segmentation model when making selections, even though incorporating information also from this source might be useful. Hybrid strategies that combine or switch between multiple query strategies were not explored in this work. Another avenue for future work is the exploration of different string similarity metrics for the representative sampling, as the Levenshtein edit distance used in this work may not yield optimal clusters.

Acknowledgments

This research has been supported by EC's Seventh Framework Programme (grant n°287678, Simple4All) and the Academy of Finland under the Finnish Centre of Excellence Program 2012–2017 (grant n°251170, COIN), LASTU Programme (grants n°256887 and 259934) and the project Fenno-Ugric Digital Citizens (grant n°270082). Computer resources within the Aalto University School of Science "Science-IT" project were used.

References

Aikio, Ante. 2005. Pohjoissaamen alkeiskurssi. Lecture material.

Baum, Leonard E. 1972. An inequality and an associated maximization technique in statistical estimation of probabilistic functions of a Markov process. *Inequalities* 3(1):1–8.

Bosch, Sonja E, Laurette Pretorius, Kholisa Podile, and Axel Fleisch. 2008. Experimental fast-tracking of morphological analysers for Nguni languages. In *LREC*.

Cohen, Jacob. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20(1):37–46.

Creutz, Mathias, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. 2007. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Transactions on Speech and Language Processing* 5(1):3:1–3:29.

Creutz, Mathias and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the Workshop on Morphological and Phonological Learning of ACL'02*, pages 21–30. Philadelphia, Pennsylvania, USA.

Creutz, Mathias and Krista Lagus. 2004. Induction of a simple morphology for highly-inflecting languages. In *Proc. 7th Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON)*, pages 43–51. Barcelona.

Creutz, Mathias and Krista Lagus. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of the AKRR'05*. Espoo, Finland.

Creutz, Mathias and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing* 4(1).

Creutz, Mathias and Krister Lindén. 2004. Morpheme segmentation gold standards for Finnish and English. Tech. Rep. A77, Publications in Computer and Information Science, Helsinki University of Technology.

Druck, Gregory, Burr Settles, and Andrew McCallum. 2009. Active learning by labeling features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 81–90. Association for Computational Linguistics.

Fishel, Mark and Harri Kirik. 2010. Linguistically motivated unsupervised segmentation for machine translation. In *LREC*.

Freund, Yoav, H Sebastian Seung, Eli Shamir, and Naftali Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine learning* 28(2-3):133–168.

Grönroos, Stig-Arne, Kristiina Jokinen, Katri Hiovain, Mikko Kurimo, and Sami Virpioja. 2015a. Low-resource active learning of North Sámi morphological segmentation. In *Proceedings of 1st International Workshop in Computational Linguistics for Uralic Languages*, pages 20–33.

Grönroos, Stig-Arne, Sami Virpioja, and Mikko Kurimo. 2015b. Tuning phrase-based segmented translation for a morphologically complex target language. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal: Association for Computational Linguistics.

Grönroos, Stig-Arne, Sami Virpioja, Peter Smit, and Mikko Kurimo. 2014. Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, pages 1177–1185. ACL.

Guyon, Isabelle, Gavin C. Cawley, Gideon Dror, and Vincent Lemaire. 2011. Results of the active learning challenge. In *Active Learning and Experimental Design workshop, In conjunction with AISTATS 2010, Sardinia, Italy, May 16, 2010*, pages 19–45.

Hammarström, Harald and Lars Borin. 2011. Unsupervised learning of morphology. *Computational Linguistics* 37(2):309–350.

Hirsimäki, Teemu, Mathias Creutz, Vesa Siivola, Mikko Kurimo, Sami Virpioja, and Janne Pylkkönen. 2006. Unlimited vocabulary speech recognition with morph language models applied to Finnish. *Computer Speech and Language* 20(4):515–541.

Jokinen, Kristiina. 2014. Open-domain interaction and online content in the sami language. In *Language Resources and Evaluation Conference*, pages 517–522.

Jokinen, Kristiina and Graham Wilcock. 2014a. Community-based resource building and data collection. In *The 4th International Workshop on Spoken Language Technologies for Under-resourced Languages (SLTU'14)*.

Jokinen, Kristiina and Graham Wilcock. 2014b. Multimodal open-domain conversations with the Nao robot. In J. Mariani, S. Rosset, M. Garnier-Rizet, and L. Devillers, eds., *Natural Interaction with Robots, Knowbots and Smartphones: Putting Spoken Dialogue Systems into Practice*, pages 213–224. Springer.

Karlsson, Fred. 1982. *Suomen kielen äänne- ja muotorakenne*. Helsinki: WSOY.

Kohonen, Oskar, Sami Virpioja, and Krista Lagus. 2010. Semi-supervised learning of concatenative morphology. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 78–86. Uppsala, Sweden: Association for Computational Linguistics.

Koskenniemi, Kimmo. 1983. *Two-level morphology: A general computational model for word-form recognition and production*. Ph.D. thesis, University of Helsinki.

Koskenniemi, Kimmo. 2008. How to build an open source morphological parser now. In *Resourceful Language Technology–Festschrift in Honor of Anna Sågvall Hein*, page 86.

Kurimo, Mikko, Mathias Creutz, and Ville Turunen. 2007. Unsupervised morpheme analysis evaluation by IR experiments – Morpho Challenge 2007. In A. Nardi and C. Peters, eds., *Working Notes for the CLEF 2007 Workshop*. CLEF. Invited paper.

Kurimo, Mikko, Sami Virpioja, and Ville T. Turunen. 2010. Overview and results of Morpho Challenge 2010. In *Proceedings of the Morpho Challenge 2010 Workshop*, pages 7–24. Espoo, Finland: Aalto University School of Science and Technology, Department of Information and Computer Science. Technical Report TKK-ICS-R37.

Levenshtein, Vladimir I. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady* 10(8):707–710.

Lewis, David D and Jason Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the eleventh international conference on machine learning*, pages 148–156.

Lewis, David D and William A Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc.

Lindén, Krister, Miikka Silfverberg, and Tommi Pirinen. 2009. Hfst tools for morphology–an efficient open-source package for construction of morphological analyzers. In *State of the Art in Computational Morphology*, pages 28–47. Springer.

McCallumzy, Andrew Kachites and Kamal Nigamy. 1998. Employing EM and pool-based active learning for text classification. In *Machine Learning: Proceedings of the Fifteenth International Conference, ICML*. Citeseer.

Nickel, Klaus Peter and Pekka Sammallahti. 2011. *Nordsamisk grammatikk*. Kárášjohka: Davvi Girji.

Oflazer, Kemal, Sergei Nirenburg, and Marjorie McShane. 2001. Bootstrapping morphological analyzers by combining human elicitation and machine learning. *Computational Linguistics* 27(1):59–85.

Poon, Hoifung, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 209–217. Association for Computational Linguistics.

Rissanen, Jorma. 1989. *Stochastic Complexity in Statistical Inquiry*, vol. 15. Singapore: World Scientific Series in Computer Science.

Ruokolainen, Teemu, Oskar Kohonen, Kairit Sirts, Stig-Arne Grönroos, Mikko Kurimo, and Sami Virpioja. 2016. A comparative study on minimally supervised morphological segmentation. *Computational Linguistics* .

Ruokolainen, Teemu, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2014. Painless semi-supervised morphological segmentation using conditional random fields. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 84–89. Gothenburg, Sweden: Association for Computational Linguistics.

Sammallahti, Pekka. 1998. *The Saami Languages: An Introduction*. Kárášjohka: Davvi Girji.

Scheffer, Tobias, Christian Decomain, and Stefan Wrobel. 2001. Active hidden markov models for information extraction. In F. Hoffmann, D. Hand, N. Adams, D. Fisher, and G. Guimaraes, eds., *Advances in Intelligent Data Analysis*, vol. 2189 of *Lecture Notes in Computer Science*, pages 309–318. Springer Berlin Heidelberg. ISBN 978-3-540-42581-6.

Settles, Burr. 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin, Madison.

Seung, H Sebastian, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM.

Sirts, Kairit and Sharon Goldwater. 2013. Minimally-supervised morphological segmentation using adaptor grammars. *TACL* 1:255–266.

Thompson, Cynthia A., Mary Elaine Califf, and Raymond J. Mooney. 1999. Active learning for natural language parsing and information extraction. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML-99)*, pages 406–414. Bled, Slovenia.

Trosterud, Trond and Heli Uibo. 2005. Consonant gradation in Estonian and Sámi: two-level solution. In *Inquiries into Words, Constraints and Contexts–Festschrift for Kimmo Koskenniemi on his 60th Birthday*, page 136. Citeseer.

Tyers, Francis M, Linda Wiechetek, and Trond Trosterud. 2009. Developing prototypes for machine translation between two Sámi languages. In *Proceedings of the 13th Annual Conf. of the EAMT*, pages 120–128.

Virpioja, Sami, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. 2013. Morfessor 2.0: Python implementation and extensions for Morfessor Baseline. Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, Department of Signal Processing and Acoustics, Aalto University, Helsinki, Finland.

Virpioja, Sami, Ville Turunen, Sebastian Spiegler, Oskar Kohonen, and Mikko Kurimo. 2011. Empirical comparison of evaluation methods for unsupervised learning of morphology. *Traitement Automatique des Langues* 52(2):45–90.

Virpioja, Sami, Jaakko J Väyrynen, Mathias Creutz, and Markus Sadeniemi. 2007. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. *Machine Translation Summit XI* 2007:491–498.

VISK. 2004. Auli Hakulinen, Maria Vilkuna, Riitta Korhonen, Vesa Koivisto, Tarja Riitta Heinonen and Irja Alho. Iso suomen kielioppi. [Online database, `http://scripta.kotus.fi/visk` referenced 7.10.2016].

Viterbi, A. J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* 13(2):260–269.

Wilcock, G., N. Laxström, J. Leinonen, P. Smit, M. Kurimo, and K. Jokinen. 2016. Towards SamiTalk: a Sami-speaking robot linked to Sami Wikipedia. In K. Jokinen and G. Wilcock, eds., *Dialogues with Social Robots: Enablements Analyses, and Evaluation*, pages 301–309. Springer.

Xu, Zhao, Kai Yu, Volker Tresp, Xiaowei Xu, and Jizhi Wang. 2003. Representative sampling for text classification using support vector machines. In F. Sebastiani, ed., *Advances in Information Retrieval*, vol. 2633 of *Lecture Notes in Computer Science*, pages 393–407. Springer Berlin Heidelberg. ISBN 978-3-540-01274-0.

Yarowsky, David and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 207–216. Association for Computational Linguistics.

Álgu-tietokanta. 2006. Kotimaisten kielten tutkimuskeskus. Sámegielaid etymologaš diehtovuođđu. [Online database, `http://kaino.kotus.fi/algu/` referenced 15.8.2015].

Publication IV

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. North Sámi morphological segmentation with low-resource semi-supervised sequence labeling. In *Proceedings of the fifth Workshop on Computational Linguistics for Uralic Languages*, Tartu, Estonia, pages 15–26, Jan 2019.

North Sámi morphological segmentation with low-resource semi-supervised sequence labeling

Stig-Arne Grönroos
stig-arne.gronroos@aalto.fi
Aalto University, Finland

Sami Virpioja
sami.virpioja@aalto.fi
Aalto University, Finland
Utopia Analytics, Finland

Mikko Kurimo
mikko.kurimo@aalto.fi
Aalto University, Finland

December 14, 2018

Abstract

Semi-supervised sequence labeling is an effective way to train a low-resource morphological segmentation system. We show that a feature set augmentation approach, which combines the strengths of generative and discriminative models, is suitable both for graphical models like conditional random field (CRF) and sequence-to-sequence neural models. We perform a comparative evaluation between three existing and one novel semi-supervised segmentation methods. All four systems are language-independent and have open-source implementations. We improve on previous best results for North Sámi morphological segmentation. We see a relative improvement in morph boundary F_1-score of 8.6% compared to using the generative Morfessor FlatCat model directly and 2.4% compared to a seq2seq baseline. Our neural sequence tagging system reaches almost the same performance as the CRF topline.

Tiivistelmä

Puoliohjattu sekvenssiluokitus on tehokas tapa opettaa morfologinen pilkontajärjestelmä kielelle, jolle on saatavilla niukasti lingvistisiä resursseja. Osoitamme, että generatiivisen mallin tuottamien piirteiden käyttäminen soveltuu paitsi graafisille malleille kuten ehdollinen satunnaiskenttä (CRF), myös sekvenssistä-sekvenssiin (seq2seq) -neuroverkkomalleille. Vertailemme kolmea olemassaolevaa ja yhtä uutta puoliohjattua menetelmää. Kaikki menetelmät ovat kieliriippumattomia, ja niille on avoimen lähdekoodin toteutus. Parannamme aikaisempia tuloksia pohjoissaamen morfologisen pilkonnan tehtävässä. Suhteelliset parannukset morfirajojen osumien F_1-mittaan ovat 8.6% verrattuna generatiiviseen Morfessor FlatCat -malliin ja 2.4% verrattuna seq2seq-verrokkimalliin. Ehdottamamme uusi neuroverkkomalli saavuttaa lähes saman tason kuin paras CRF-malli.

Proceedings of the fifth Workshop on Computational Linguistics for Uralic Languages, pages 15–26,
Tartu, Estonia, January 7 - January 8, 2019.

1 Introduction

Subword models have enjoyed recent success in many natural language processing (NLP) tasks, such as machine translation (Sennrich et al., 2015) and automatic speech recognition (Smit et al., 2017). Uralic languages have rich morphological structure, making morphological segmentation particularly useful for these languages. While rule-based morphological segmentation systems can achieve high quality, the large amount of human effort needed makes the approach problematic for low-resource languages. As a fast, cheap and effective alternative, data-driven segmentation can be learned based on a very small amount of human annotator effort. Using active learning, as little as some hundreds of annotated word types can be enough (Grönroos et al., 2016).

Adopting neural methods has lead to a large performance gain for many NLP tasks. However, neural networks are typically data-hungry, reducing their applicability to low-resource languages. Most research has focused on high-resource languages and large data sets, while the search for new approaches to make neural methods applicable to small data has only recently gained attention. For example, the workshop Deep Learning Approaches for Low-Resource NLP (DeepLo[1]) was arranged first time in the year of writing. Neural methods have met with success in high-resource morphological segmentation (e.g. Wang et al., 2016). We are interested to see if data-hungry neural network models are applicable to segmentation in low-resource settings, in this case for the Uralic language North Sámi.

Neural sequence-to-sequence (seq2seq) models are a very versatile tool for NLP, and are used in state of the art methods for a wide variety of tasks, such as text summarization (Nallapati et al., 2016) and speech synthesis (Wang et al., 2017). Seq2seq methods are easy to apply, as you can often take e.g. existing neural machine translation software and train it with appropriately preprocessed data. Kann et al. (2018) apply the seq2seq model for low-resource morphological segmentation.

However, arbitrary length sequence-to-sequence transduction is not the optimal formulation for the task of morphological surface segmentation. We return to formulating it as a a sequence tagging problem instead, and show that this can be implemented with minor modifications to an open source translation system.

Moreover, we show that the semi-supervised training approach of Ruokolainen et al. (2014) using feature set augmentation can also be applied to neural networks to effectively leverage large unannotated data.

2 Morphological processing tasks

There are several related morphological tasks that can be described as mapping from one sequence to another. Morphological segmentation is the task of splitting words into morphemes, meaning-bearing sub-word units. In morphological *surface* segmentation, the word w is segmented into a sequence of surface morphs, substrings whose concatenation is the word w.

$$\text{e.g.} \quad achievability \mapsto achiev \circ abil \circ ity$$

Canonical morphological segmentation (Kann et al., 2016) instead yields a sequence of standardized segments. The aim is to undo morphological processes that result in

[1] https://sites.google.com/view/deeplo18/home

allomorphs, i.e. different surface morphs corresponding to the same meaning.

$$w \mapsto y; \quad w \in \Sigma^*, y \in (\Sigma \cup \{\circ\})^*$$

$$\text{e.g.} \quad achievability \mapsto achieve \circ able \circ ity$$

where Σ is the alphabet of the language, and $\circ$ is the boundary marker.

Morphological analysis yields the lemma and tags representing the morphological properties of a word.

$$w \mapsto yt; \quad w, y \in \Sigma^*, t \in \tau^*$$

$$\text{e.g.} \quad took \mapsto take \text{ PAST}$$

where τ is the set of morphological tags.

Two related morphological tasks are reinflection and lemmatization. In *morphological reinflection* (see e.g. Cotterell et al., 2016), one or more inflected forms are given to identify the lexeme, together with the tags identifying the desired inflection. The task is to produce the correctly inflected surface form of the lexeme.

$$wt \mapsto y; \quad w, y \in \Sigma^*, t \in \tau^*$$

$$\text{e.g.} \quad taken \text{ PAST} \mapsto took$$

In *lemmatization*, the input is an inflected form and the output is the lemma.

$$w \mapsto y; \quad w, y \in \Sigma^*$$

$$\text{e.g.} \quad better \mapsto good$$

Morphological surface segmentation can be formulated in the same way as canonical segmentation, by just allowing the mapping to canonical segments to be the identity. However, this formulation fails to capture the fact that the segments must concatenate back to the surface form. The model is allowed to predict any symbol from its output vocabulary, although only two symbols are valid at any given timestep: the boundary symbol or the actual next character. If the labeled set for supervised training is small, the model may struggle with learning to copy the correct characters. Kann et al. (2018) address this problem by a multi-task training approach where the auxiliary task consists of reconstructing strings in a sequence auto-encoder setting. The strings to be reconstructed can be actual words or even random noise.

Surface segmentation can alternatively be formulated as structured classification

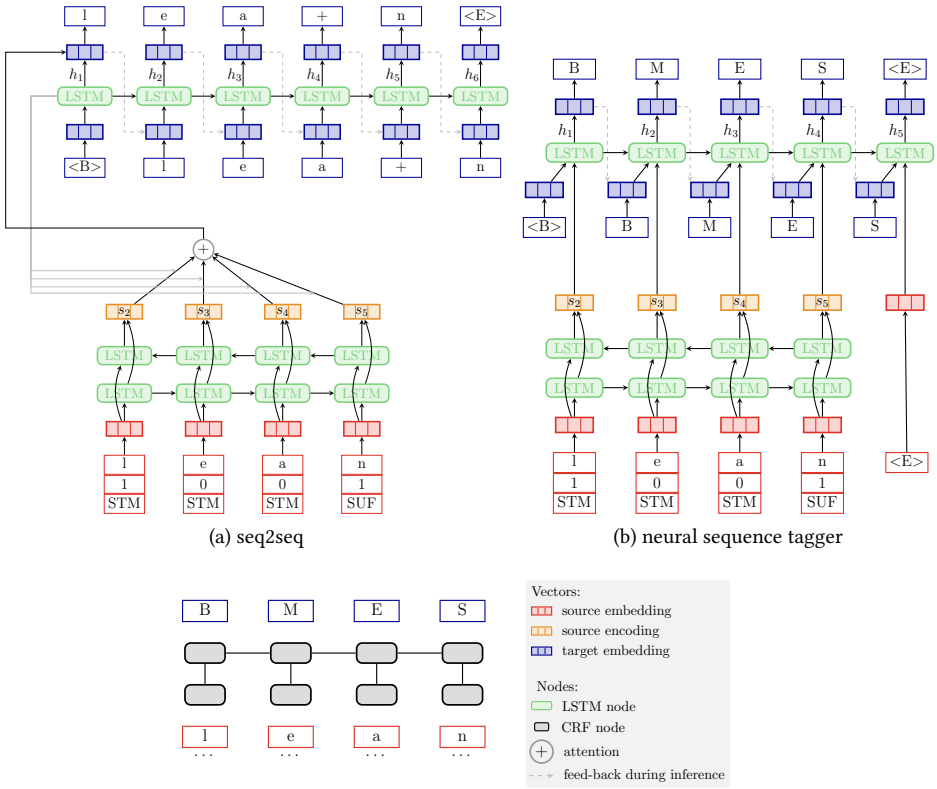
$$w \mapsto y; \quad w \in \Sigma^k, y \in \Omega^k, k \in \mathbb{N}$$

$$\text{e.g.} \quad uses \mapsto BMES$$

where Ω is the segmentation tag set. Note that there is no need to generate characters from the original alphabet, instead a small tag set Ω is used. The fact that the sequence of boundary decisions is of the same length k as the input has also been made explicit.

Different tag sets Ω can be used for segmentation. The minimal sets only include two labels: BM/ME (used e.g. by Green and DeNero, 2012). Either the beginning (B) or end (E) of segments is distinguished from non-boundary time-steps in the middle (M). A more fine-grained approach BMES[2] (used e.g. by Ruokolainen et al., 2014) uses

[2]Also known as BIES, where I stands for internal.



(a) seq2seq (b) neural sequence tagger

(c) conditional random field

Figure 1: Model architectures. To prevent the figure from becoming too large, the seq2seq model is drawn with only one LSTM layer in both encoder and decoder. The attention is only shown for the fist time step.

four labels. In addition to marking both beginning and end of segments, a special label is used for single-character (S) segments.

Morphological analysis or canonical segmentation resolve ambiguity, and are more informative than surface segmentation. Learning to resolve such ambiguity is a more challenging task to learn than surface segmentation. Surface segmentation may be preferred over the other tasks e.g. when used in an application that needs to generate text in a morphologically complex language, such as when it is the target language in machine translation. If surface segments are generated, the final surface form is easily recovered through concatenation.

To summarize, arbitrary-length sequence transduction is a formulation well suited for many morphological tasks. Morphological surface segmentation is an exception, being more appropriately formulated as sequence tagging.

3 Models for semi-supervised segmentation

Our semi-supervised training follows the approach of Ruokolainen et al. (2014). The training data consists of a large unlabeled set, and a smaller labeled training set. The labeled training set is further divided into two parts. A generative model, in our

Factor	Emb																	
character	350	b	i	e	b	m	o	r	á	h	k	a	d	e	a	m	i	s
boundary	10	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1
category	10	M	M	M	M	M	M	M	M	M	M	M	M	f	f	f	f	f

Table 1: Example input factors with embedding dimension. The example word biebmoráhkadeamis is segmented as biebmo/STM ráhkad/STM eami/SUF s/SUF. Stem (STM) is abbreviated M, and suffix (SUF) is f.

case Morfessor FlatCat, is trained in a semi-supervised fashion using the first part of the labeled training set. The words in the second part of the labeled training set are segmented using the generative model. Now these words are associated with two segmentations: predicted and gold standard. A discriminative model is then trained on the second part of the labeled training set. The predictions of the generative model are fed into the discriminative model as augmented features. The gold standard segmentation is used as the target sequence.

At decoding time a two-step procedure is used: first the features for the desired words are produced using the generative model. The final segmentation can then be decoded from the discriminative model.

The idea is that the features from the generative model allow the statistical patterns found in the large unannotated data to be exploited. At the same time, the capacity of the discriminative model is freed for learning to determine when the generative model's predictions are reliable, in essence to only correct its mistakes.

3.1 Morfessor FlatCat

We produce the features for our semi-supervised training using Morfessor FlatCat (Grönroos et al., 2014). Morfessor FlatCat is a generative probabilistic method for learning morphological segmentations. It uses a prior over morph lexicons inspired by the Minimum Description Length principle (Rissanen, 1989). Morfessor FlatCat applies a simple Hidden Markov model for morphotactics, providing morph category tags (stem, prefix, suffix) in addition to the segmentation. The segmentations are more consistent compared to Morfessor Baseline, particularly when splitting compound words.

Morfessor FlatCat produces morph category labels in addition to the segmentation decisions. These labels can also be used as features. An example of the resulting 3-factor input is shown in Table 1.

3.2 Sequence-to-sequence

Our sequence-to-sequence (seq2seq) baseline model follows Kann et al. (2018) with some minor modifications. It is based on the encoder-decoder with attention (Bahdanau et al., 2014). The encoder is a 2-layer bidirectional Long Short-Term Memory (LSTM) layer (Hochreiter and Schmidhuber, 1997), while the decoder is a 2-layer LSTM. The model is trained on the character level.

Figure 1a shows the basic structure of the architecture. For simplicity a single layer is shown for both encoder and decoder.

3.3 Conditional random fields

Conditional random fields (CRF) are discriminative structured classification models for sequential tagging and segmentation (Lafferty et al., 2001). They are expressed as undirected probabilistic graphical models. Figure 1c shows the model structure. CRFs can be seen as generalizing the log-linear classifier to structured outputs. They bear a structural resemblance to hidden Markov models, while relaxing the assumption of the observations being conditionally independent given the labels.

We use the implementation of linear-chain CRFs by Ruokolainen et al. (2014)[3].

3.4 Neural sequence tagger

The encoder is a standard single-layer bidirectional LSTM. The decoder is a single-layer LSTM, which takes as input at time t the concatenation of the encoder output at time t and an embedding of the predicted label at $t-1$. There is no attention mechanism. However, the time-dependent connection to the encoder could be described as a hard-coded diagonal monotonic attention that always moves one step forward. The architecture can be seen in Figure 1b.

The most simple fixed-length decoding strategy is to forego structured prediction and instead make a prediction at each time-step based only on the encoder output s_t. The prediction at each time-step is then conditionally independent given the hidden states. We choose to instead feed the previous decision back in, causing a left-to-right dependence on previous decisions.

The proposed model has only 5% of the number of parameters of the seq2seq model (469 805 versus 8 820 037). The proposed model requires no attention mechanism, and the target vocabulary is much smaller. We also found that the optimal network size in terms of number of layers and vector dimensions was smaller.

We use factored input for the additional features. The FlatCat segmentation decision and morph category label are independently embedded. These factor embeddings are concatenated to the character embedding.

Because our human annotations include the category labels, we use a simple target-side multi-task setup to predict them in addition the the segmentation boundaries. The output vocabulary is extended to cover all combinations of segmentation decision and category label. Because our data set contains two morph categories, STM and SUF, this only increases the size of the output vocabulary from 5 (BMES + end symbol) to 10.

We use a modified beam search to ensure that the output sequence is of the correct length. This is achieved by manipulating the probability of the end symbol, setting it to zero if the sequence is still too short and to one when the correct length is reached.

The system is implemented by extending OpenNMT (Klein et al., 2017). Our implementation is open source[4].

4 North Sámi

North Sámi (davvisámegiella) is a Finno-Ugric language, spoken in the northern parts of Norway, Sweden, Finland and Russia. With around 20 000 speakers, it is biggest of the nine Sámi languages.

[3]Available from `http://users.ics.tkk.fi/tpruokol/software/crfs_morph.zip`

[4]Available from `https://github.com/Waino/OpenNMT-py/tree/same_length_decoder`

Purpose	Subset	Component	Word types	Labels
Training	Unlabeled	FlatCat	691190	No
Training	Feature train	FlatCat	200	Yes
	Main train	System	844	Yes
Development		Both	199	Yes
Testing			796	Yes

Table 2: Subdivision of data sets, with size in word types. The component column indicates which components use the data during training.

North Sámi is a morphologically complex language, featuring both rich inflection, derivation and productive compounding. It has complicated although regular morphophonological variation. Compounds are written together without an intermediary space. For example nállošalbmái ("into the eye of the needle"), could be segmented as nállo ○ šalbmá ○ i.

The morphology of Sámi languages has been modeled using finite state methods (Trosterud and Uibo, 2005; Lindén et al., 2009). The Giellatekno research lab[5] provides rule-based morphological analyzers both for individual word forms and running text, in addition to miscellaneous other resources such as wordlists and translation tools. A morphological analyzer is not a direct replacement for morphological segmentation, as there is no trivial way to map from analysis to segmentation. In addition to this, rule-based analyzers are always limited in their coverage of the vocabulary.

For an overview into the Giellatekno/Divvun and Apertium projects, including their work on Sámi languages, see Moshagen et al. (2014).

5 Data

We use version 2 of the data set collected by (Grönroos et al., 2015; Grönroos et al., 2016) as the labeled data, and as unlabeled data a word list extracted from *Den samiske tekstbanken* corpus[6].

The labeled data contains words annotated for morphological segmentation with morph category labels. The annotations were produced by a single Sámi scholar, who is not a native speaker of Sámi. In total 2311 annotated words were available. The development and test sets contain randomly selected words. The training set set of 1044 annotations is the union of 500 randomly selected words and and 597 using different active learning approaches. There was some overlap in the sets. Due to the active learning, it should be assumed that the data set is more informative than a randomly selected data set of the same size.

Table 2 shows how the data was subdivided. The unlabeled data, the development set and the test set are the same as in Grönroos et al. (2016). To produce the two labeled training sets, we first combined the labeled training data collected with different methods. From this set, 200 word types were randomly selected for semi-supervised training of Morfessor FlatCat, and the remaining 844 were used for training the dis-

[5] `http://giellatekno.uit.no/`

[6] Provided by UiT, The Arctic University of Norway.

criminative system. These two labeled data sets must be disjoint to avoid the system overestimating the reliability of the FlatCat output.

6 Training details

Tuning of FlatCat was performed following Grönroos et al. (2016). The corpus likelihood weight α was set to 1.4. The value for the annotation likelihood weight β was set using a heuristic formula optimized for Finnish:

$$\log \beta = 1.9 + 0.8 \log |\boldsymbol{D}| - 0.6 \log |\boldsymbol{A}|, \tag{1}$$

where $|\boldsymbol{D}|$ and $|\boldsymbol{A}|$ are the numbers of word types in the unannotated and annotated training data sets, respectively. Using this formula resulted in setting β to 13000. Perplexity threshold for suffixes was set to 40. For prefixes we used a high threshold (999999) to prevent the model from using them, as there are no prefixes in North Sámi.

The neural networks were trained using SGD with learning rate 1.0. Gradient norm was clipped to 5.0. Batch size was set to 64 words. Embeddings were dropped out with probability 0.3. Models were trained for at most 5000 steps, and evaluated for early stopping every 250 steps.

For the neural sequence tagger, the embedding size was 350 for characters and 10 for other input factors, and 10 for target embeddings. The encoder single bi-LSTM layer size was set to 150.

All neural network results are the average of 5 independent runs with different seeds.

7 Evaluation

The segmentations generated by the model are evaluated by comparison with annotated morph boundaries using *boundary precision*, *boundary recall*, and *boundary* F_1*-score* (see e.g., Virpioja et al., 2011). The boundary F_1-score equals the harmonic mean of precision (the percentage of correctly assigned boundaries with respect to all assigned boundaries) and recall (the percentage of correctly assigned boundaries with respect to the reference boundaries).

$$\text{Precision} = \frac{\#(\text{correct})}{\#(\text{proposed})}; \qquad \text{Recall} = \frac{\#(\text{correct})}{\#(\text{reference})} \tag{2}$$

Precision and recall are calculated using macro-averages over the words in the test set. In the case that a word has more than one annotated segmentation, we take the one that gives the highest score.

In order to evaluate boundary precision and recall, a valid segmentation is needed for all words in the test set. The seq2seq model can fail to output a valid segmentation, in which case we replace the output with the input without any segmentation boundaries. To include an evaluation without this source of error we also report word type level accuracy. A word in the test set is counted as correct if all boundary decisions are correct. Output that does not concatenate back to the input word is treated as incorrect.

system	Pre	Rec	F_1	w-acc
FlatCat (200 words)	78.20	77.60	77.90	57.20
Seq2seq (s)	86.94	78.62	82.54	64.60
NST (s)	83.26	83.92	83.58	69.12
CRF (s)	**87.70**	83.30	85.40	69.30
FlatCat (full)	74.30	84.10	78.90	61.80
Seq2seq (ss)	87.66	80.16	83.72	68.36
NST (ss)	84.28	**85.58**	84.94	71.02
CRF (ss)	86.30	85.20	**85.70**	**71.10**

Table 3: Results on the test set. Boundary precision (Pre), recall (Rec), and F_1-scores, together with word-type level accuracy (w-acc). NST is short for neural sequence tagger. FlatCat (200 words) shows the performance of the FlatCat system used to produce the input features. FlatCat (full) line shows FlatCat trained using the full training set. Fully supervised models, i.e. without using FlatCat features, are marked (s). Semi-supervised models are marked (ss).

	STM	STM + STM			STM + SUF			STM + SUF + SUF		
	Pre	Pre	Rec	F_1	Pre	Rec	F_1	Pre	Rec	F_1
FlatCat (200)	71.50	78.50	70.90	74.50	77.60	69.90	73.50	78.90	56.70	66.00
Seq2seq (s)	82.02	89.82	66.54	76.08	88.22	74.78	80.94	81.44	56.10	66.32
NST (s)	76.74	83.84	78.20	80.90	86.04	79.82	82.82	80.30	58.34	67.58
CRF (s)	79.80	95.50	60.00	73.70	**89.40**	**82.70**	**85.90**	83.30	62.70	71.60
FlatCat (full)	62.70	82.50	**92.70**	87.30	76.70	76.40	76.60	72.90	61.90	67.00
Seq2seq (ss)	**82.46**	91.08	68.00	77.80	88.46	76.46	82.00	84.74	57.60	68.56
NST (ss)	78.78	87.90	86.56	87.20	85.28	80.12	82.60	78.20	59.54	67.58
CRF (ss)	77.20	**96.40**	85.50	**90.60**	86.60	79.40	82.80	**88.60**	**67.90**	**76.90**

Table 4: Boundary precision (Pre), recall (Rec), and F_1-scores for different subsets of the evaluation data. NST stands for Neural sequence tagger. (s) stands for fully supervised, (ss) for semi-supervised.

8 Results

Table 3 shows results on the full test set. The semi-supervised CRF shows the best performance both according to F_1-score and word-type level accuracy. Semi-supervised seq2seq has high precision but low recall, indicating under-segmentation. The neural sequence tagger shows the opposite behavior, with the highest recall.

All semi-supervised methods improve on the quality of the semi-supervised FlatCat trained on 200 annotated words which is used as input features. All three discriminative methods also outperform FlatCat trained on the whole training set, on F_1 and accuracy. All three semi-supervised methods outperform their fully supervised variants. These results show that two-step training is preferable over using only Morfessor FlatCat or one of the discrinative methods.

The seq2seq model frequently fails to output a valid segmentation, either generating incorrect characters, stopping too early, or getting stuck repeating a pattern of characters. For 10.7% of the test set, the seq2seq output does not concatenate back to the input word.

Table 4 shows results for subsets of the evaluation data. The subsets include all words were the gold standard category labels follow a particular pattern: No internal structure (STM), uninflected compound (STM+STM), single-suffix inflected word (STM+SUF) and two-suffix inflected word (STM+SUF+SUF).

The seq2seq model has the best performance for the STM-pattern. This is only partly explained by the bias towards not segmenting at all caused by the replacement procedure for the invalid outputs.

The seq2seq model has high precision for all category patterns. Fully supervised CRF has superior precision and recall for the STM+SUF pattern, while semi-supervised CRF is superior for the STM+SUF+SUF pattern. CRF is good at modeling the boundaries of suffixes. Adding the FlatCat features improves the modeling of the boundary between multiple suffixes, while slightly deteriorating the modeling of the boundary between stem and suffix. The left-to-right decoding is a possible explanation for the weaker performance of the neural sequence tagger on the STM+SUF+SUF pattern. Fully supervised CRF is poor at splitting compound words, evidenced by the low recall for the STM+STM pattern. This deficiency is effectively alleviated by the addition of the FlatCat features.

The neural sequence tagger is good at modeling the ends of stems, indicated by high recall on the STM+STM and STM+SUF patterns.

9 Conclusions and future work

Semi-supervised sequence labeling is an effective way to train a low-resource morphological segmentation system. We recommend training a CRF sequence tagger using a Morfessor FlatCat-based feature set augmentation approach. This setup achieves a morph boundary F_1-score of 85.70, improving on previous best results for North Sámi morphological segmentation. Our neural sequence tagging system reaches almost the same word-type level accuracy as the CRF system, while having better morph boundary recall.

The bidirectional LSTM-CRF model (Huang et al., 2015) uses the power of a recurrent neural network to combine contextual features, and stacks a CRF on top for sequence level inference. The performance of this architecture on the North Sámi morphological segmentation task should be explored in future work.

Acknowledgments

This research has been supported by the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 780069. Computer resources within the Aalto University School of Science "Science-IT" project were used.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR15*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The sigmorphon 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. pages 10–22.

Spence Green and John DeNero. 2012. A class-based agreement model for generating accurately inflected translations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 146–155.

Stig-Arne Grönroos, Katri Hiovain, Peter Smit, Ilona Erika Rauhala, Päivi Kristiina Jokinen, Mikko Kurimo, and Sami Virpioja. 2016. Low-resource active learning of morphological segmentation. *Northern European Journal of Language Technology* .

Stig-Arne Grönroos, Kristiina Jokinen, Katri Hiovain, Mikko Kurimo, and Sami Virpioja. 2015. Low-resource active learning of North Sámi morphological segmentation. In *Proceedings of 1st International Workshop in Computational Linguistics for Uralic Languages*. pages 20–33.

Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. 2014. Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*. Association for Computational Linguistics, Dublin, Ireland, pages 1177–1185.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR arXiv:1508.01991* .

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2016. Neural morphological analysis: Encoding-decoding canonical segments. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 961–967.

Katharina Kann, Manuel Mager, Ivan Meza, and Hinrich Schütze. 2018. Fortification of neural morphological segmentation models for polysynthetic minimal-resource languages. In *Proceedings of NAACL 2018*. Association for Computational Linguistics, New Orleans, Louisiana, USA.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: open-source toolkit for neural machine translation. In *Proc. ACL*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*. pages 282–289.

Krister Lindén, Miikka Silfverberg, and Tommi Pirinen. 2009. HFST tools for morphology–an efficient open-source package for construction of morphological analyzers. In *State of the Art in Computational Morphology*, Springer, pages 28–47.

Sjur Moshagen, Jack Rueter, Tommi Pirinen, Trond Trosterud, and Francis M. Tyers. 2014. Open-source infrastructures for collaborative work on under-resourced languages. In *Collaboration and Computing for Under-Resourced Languages in the Linked Open Data Era. LREC 2014.*. pages 71–77.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *in Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. pages 280–290.

Jorma Rissanen. 1989. *Stochastic Complexity in Statistical Inquiry*, volume 15. World Scientific Series in Computer Science, Singapore.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2014. Painless semi-supervised morphological segmentation using conditional random fields. In *European Chapter of the Association for Computational Linguistics (EACL)*. Association for Computational Linguistics, Gothenburg, Sweden, pages 84–89.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. In *ACL16*.

Peter Smit, Sami Virpioja, Mikko Kurimo, et al. 2017. Improved subword modeling for WFST-based speech recognition. In *In INTERSPEECH 2017–18th Annual Conference of the International Speech Communication Association.*.

Trond Trosterud and Heli Uibo. 2005. Consonant gradation in Estonian and Sámi: two-level solution. In *Inquiries into Words, Constraints and Contexts–Festschrift for Kimmo Koskenniemi on his 60th Birthday*, Citeseer, page 136.

Sami Virpioja, Ville T Turunen, Sebastian Spiegler, Oskar Kohonen, and Mikko Kurimo. 2011. Empirical comparison of evaluation methods for unsupervised learning of morphology. *Traitement Automatique des Langues* 52(2):45–90.

Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. 2016. Morphological segmentation with window LSTM neural networks. In *AAAI*. pages 2842–2848.

Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Q Le, Y Agiomyrgiannakis, R Clark, and R. A. Saurous. 2017. Tacotron: Towards end-to-end speech synthesis. In *Proc. Interspeech*. pages 4006–4010.

Publication V

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. Morfessor EM+Prune: Improved subword segmentation with expectation maximization and pruning. In *Proceedings of the 12th Language Resources and Evaluation Conference*, Marseilles, France, 2020, pages 3944–3953, May 2020.

Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 3944–3953
Marseille, 11–16 May 2020

Morfessor EM+Prune: Improved Subword Segmentation with Expectation Maximization and Pruning

Stig-Arne Grönroos[1], **Sami Virpioja**[2], **Mikko Kurimo**[1]
[1]Department of Signal Processing and Acoustics, Aalto University, Finland
[2]Department of Digital Humanities, University of Helsinki, Finland
{stig-arne.gronroos,mikko.kurimo}@aalto.fi, sami.virpioja@helsinki.fi

Abstract

Data-driven segmentation of words into subword units has been used in various natural language processing applications such as automatic speech recognition and statistical machine translation for almost 20 years. Recently it has became more widely adopted, as models based on deep neural networks often benefit from subword units even for morphologically simpler languages. In this paper, we discuss and compare training algorithms for a unigram subword model, based on the Expectation Maximization algorithm and lexicon pruning. Using English, Finnish, North Sami, and Turkish data sets, we show that this approach is able to find better solutions to the optimization problem defined by the Morfessor Baseline model than its original recursive training algorithm. The improved optimization also leads to higher morphological segmentation accuracy when compared to a linguistic gold standard. We publish implementations of the new algorithms in the widely-used Morfessor software package.

Keywords: Morphology, Statistical and Machine Learning Methods, Language Modelling, Unsupervised learning, Tools, Less-Resourced/Endangered Languages

1. Introduction

Subword segmentation has become a standard preprocessing step in many neural approaches to natural language processing (NLP) tasks, e.g Neural Machine Translation (NMT) (Sennrich et al., 2015) and Automatic Speech Recognition (ASR) (Smit et al., 2017). Word level modeling suffers from sparse statistics, issues with Out-of-Vocabulary (OOV) words, and heavy computational cost due to a large vocabulary. Word level modeling is particularly unsuitable for morphologically rich languages, but subwords are commonly used for other languages as well. Subword segmentation is best suited for languages with agglutinative morphology.

While rule-based morphological segmentation systems can achieve high quality, the large amount of human effort needed makes the approach problematic, particularly for low-resource languages. The systems are language dependent, necessitating use of multiple tools in multilingual setups. As a fast, cheap and effective alternative, data-driven segmentation can be learned in a completely unsupervised manner from raw corpora.

Unsupervised morphological segmentation saw much research interest until the early 2010's; for a survey on the methods, see Hammarström and Borin (2011). Semi-supervised segmentation with already small amounts of annotated training data was found to improve the accuracy significantly when compared to a linguistic segmentation; see Ruokolainen et al. (2016) for a survey. While this line of research has been continued in supervised and more grammatically oriented tasks (Cotterell et al., 2017), the more recent work on unsupervised segmentation is less focused on approximating a linguistically motivated segmentation. Instead, the aim has been to tune subword segmentations for particular applications. For example, the simple substitution dictionary based Byte Pair Encoding segmentation algorithm (Gage, 1994), first proposed for NMT by Sennrich et al. (2015), has become a standard in the field. Especially in the case of multilingual models, training a single language-independent subword segmentation method is preferable to linguistic segmentation (Arivazhagan et al., 2019).

In this study, we compare three existing and one novel subword segmentation method, all sharing the use of a unigram language model in a generative modeling framework. The previously published methods are Morfessor Baseline (Creutz and Lagus, 2002), Greedy Unigram Likelihood (Varjokallio et al., 2013), and SentencePiece (Kudo, 2018). The new Morfessor variant proposed in this work is called Morfessor EM+Prune.

The contributions of this article are

(i) a better training algorithm for Morfessor Baseline, with reduction of search error during training, and improved segmentation quality for English, Finnish and Turkish;

(ii) comparing four similar segmentation methods, including a close look at the SentencePiece reference implementation, highlighting details omitted from the original article (Kudo, 2018);

(iii) and showing that the proposed Morfessor EM+Prune with particular hyper-parameters yields SentencePiece.

1.1. Morphological Segmentation with Unigram Language Models

Morphological surface segmentation is the task of splitting words into morphs, the surface forms of meaning-bearing sub-word units, morphemes. The concatena-

	Morfessor BL	Greedy Unigram	SentencePiece	Morfessor EM+Prune
Model	Unigram LM	Unigram LM	Unigram LM	Unigram LM
Cost function	MAP	ML	MAP	MAP
Prior	MDL	–	DP	MDL+DP
Training algorithm	Local search	EM+Prune	EM+Prune	EM+Prune
Initialization	Words	Seed lexicon	Seed lexicon	Seed lexicon
EM variant	–	Lateen-EM once	EM	EM / Lateen-EM
Stopping criterion				
Cost change threshold	✓	✓	–	✓
Target lexicon size	Approximate	✓	✓	✓
N-best decoding	✓	–	✓	✓
Sampling decoding	–	–	✓	✓
Count dampening	✓	–	–	✓
Semi-supervised	✓	–	–	✓
Requires pretokenization	✓	✓	–	✓
Reference implementation	Python	C++	C++	Python

Table 1: Comparison of subword segmentation methods applying a unigram language model.

tion of the morphs is the word, e.g.

$$reliability \mapsto reli + abil + ity$$

Probabilistic generative methods for morphological segmentation model the probability $P(\boldsymbol{s})$ of generating a sequence of morphs (a word, sentence or corpus) $\boldsymbol{s} = [m_0, \ldots, m_N]$, as opposed to discriminative methods that model the conditional probability of the segmentation boundaries given the unsegmented data.

This study focuses on segmentation methods applying a ***unigram language model***. In the unigram language model, an assumption is made that the morphs in a word occur independently of each other. Alternatively stated, it is a zero-order (memoryless) Markov model, generalized so that one observation can cover multiple characters. The probability of a sequence of morphs decomposes into the product of the probabilities of the morphs of which it consists.

$$P_{\boldsymbol{\theta}}(\boldsymbol{s}) = \prod_{i=1}^{N} P_{\boldsymbol{\theta}}(m_i) \tag{1}$$

The Expectation Maximization (EM) algorithm (Dempster et al., 1977) is an iterative algorithm for finding Maximum Likelihood (ML) or Maximum a Posteriori (MAP) estimates for parameters in models with latent variables. The EM algorithm consists of two steps. In the E-step (2), the expected value of the complete data likelihood including the latent variable is taken, and in the M-step (3), the parameters are updated to maximize the expected value of the E-step:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(i-1)}) = \int_{\boldsymbol{y}} \log P(\boldsymbol{D}, \boldsymbol{y} \mid \boldsymbol{\theta}) P(\boldsymbol{y} \mid \boldsymbol{D}, \boldsymbol{\theta}^{(i-1)}) d\boldsymbol{y} \tag{2}$$

$$\boldsymbol{\theta}^{i} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(i-1)}). \tag{3}$$

When applied to a (hidden) Markov model, EM is called the forward-backward algorithm. Using instead the related Viterbi algorithm (Viterbi, 1967) is sometimes referred to as *hard-EM*.[1] Spitkovsky et al. (2011) present lateen-EM, a hybrid variant in which EM and Viterbi optimization are alternated.

Virpioja (2012, Section 6.4.1.3) discusses the challenges of applying EM to learning of generative morphology. Jointly optimizing both the morph lexicon and the parameters for the morphs is intractable. If, like in Morfessor Baseline, the cost function is discontinuous when morphs are added or removed from the lexicon, there is no closed form solution to the M-step. With ML estimates for morph probabilities, EM can neither add nor remove morphs from the lexicon, because it can neither change a zero probability to nonzero nor vice versa.

One solution to this challenge is to apply local search. Starting from the current best estimate for the parameters, small search steps are tried to explore near-lying parameter configurations. The choice that yields the lowest cost is selected as the new parameters. Greedy local search often gets stuck in local minima. Even if there are parameters yielding a better cost, the search may not find them, causing search error. The error remaining at the parameters with globally optimal cost is the model error.

Another solution is to combine EM with pruning (EM+Prune). The methods based on pruning begin with a seed lexicon, which is then iteratively pruned until a stopping condition is reached. Subwords cannot be added to the lexicon after initialization. As a consequence, proper initialization is important, and the methods should not prune too aggressively without reestimating parameters, as pruning decisions cannot be backtracked. For this reason, EM+Prune methods

[1] An analogy can be drawn to clustering using k-means, which yields a hard assignment of data points to clusters, and using EM for clustering with a Gaussian Mixture Model (GMM), where the assignment is soft.

proceed iteratively, only pruning subwords up to a predefined iteration pruning quota, e.g. removing at most 20% of the remaining lexicon at a time.

2. Related Work

In this section we review three previously published segmentation methods that apply a unigram language model. Table 1 summarizes the differences between these methods.

2.1. Morfessor Baseline

Morfessor is a family of generative models for unsupervised morphology induction (Creutz and Lagus, 2007). Here, consider the Morfessor 2.0 implementation (Virpioja et al., 2013) of Morfessor Baseline method (Creutz and Lagus, 2002).

A point estimate for the model parameters $\hat{\boldsymbol{\theta}}$ is found using MAP estimation with a Minimum Description Length (MDL) (Rissanen, 1989) inspired prior that favors lexicons containing fewer, shorter morphs. The MAP estimate yields a two-part cost function, consisting of a prior (the lexicon cost) and likelihood (the corpus cost). The model can be tuned using the hyperparameter α, which is a weight applied to the likelihood (Kohonen et al., 2010):

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \{ -\log \overbrace{P(\boldsymbol{\theta})}^{\text{prior}} \; -\alpha \log \overbrace{P(\boldsymbol{D} \,|\, \boldsymbol{\theta})}^{\text{likelihood}} \} \qquad (4)$$

The α parameter controls the overall amount of segmentation, with higher values increasing the weight of each emitted morph in the corpus (leading to less segmentation), and lower values giving a relatively larger weight to a small lexicon (more segmentation).

The prior can be further divided into two parts: the prior for the morph form properties and the usage properties. The form properties encode the string representation of the morphs, while the usage properties encode their frequencies. Morfessor Baseline applies a non-informative prior for the distribution of the morph frequencies. It is derived using combinatorics from the number of ways that the total token count ν can be divided among the μ lexicon items:

$$P(\tau_1, \ldots, \tau_\mu \,|\, \mu, \nu) = 1 / \binom{\nu - 1}{\mu - 1}. \qquad (5)$$

Morfessor Baseline is initialized with a seed lexicon of whole words. The Morfessor Baseline training algorithm is a greedy local search. During training, in addition to storing the model parameters, the current best segmentation for the corpus is stored in a graph structure. The segmentation is iteratively refined, by looping over all the words in the corpus in a random order and resegmenting them. The resegmentation is applied by recursive binary splitting, leading to changes in other words that share intermediary units with the word currently being resegmented. The search converges to a local optimum, and is known to be sensitive to the initialization (Virpioja et al., 2013).

In the Morfessor 2.0 implementation, the likelihood weight hyper-parameter α is set either with a grid search using the best evaluation score on a held-out development set, or by applying an approximate automatic tuning procedure based on a heuristic guess of which direction the α parameter should be adjusted.

2.2. Greedy Unigram Likelihood

Varjokallio et al. (2013) presents a subword segmentation method, particularly designed for use in ASR. It applies greedy pruning based on unigram likelihood. The seed lexicon is constructed by enumerating all substrings from a list of common words, up to a specified maximum length. Pruning proceeds in two phases, which the authors call *initialization* and *pruning*.

In the first phase, a character-level language model is trained. The initial probabilities of the subwords are computed using the language model. The probabilities are refined by EM, followed by hard-EM. During the hard-EM, frequency based pruning of subwords begins. In the second phase, hard-EM is used for parameter estimation. At the end of each iteration, the least frequent subwords are selected as candidates for pruning. For each candidate subword, the change in likelihood when removing the subword is estimated by resegmenting all words in which the subword occurs. After each pruned subword, the parameters of the model are updated. Pruning ends when the goal lexicon size is reached or the change in likelihood no longer exceeds a given threshold.

2.3. SentencePiece

SentencePiece (Kudo and Richardson, 2018; Kudo, 2018) is a subword segmentation method aimed for use in any NLP system, particularly NMT. One of its design goals is use in multilingual systems.

Although (Kudo, 2018) implies a use of maximum likelihood estimation, the reference implementation[2] uses the implicit Dirichlet Process prior called Bayesian EM (Liang and Klein, 2007). In the M-step, the count normalization is modified to

$$P(z) = \frac{\exp(\Psi(C_z))}{\exp(\Psi(\sum_{z'} C_{z'}))} \qquad (6)$$

where Ψ is the digamma function.

The seed lexicon is simply the e.g. one million most frequent substrings. SentencePiece uses an EM+Prune training algorithm. Each iteration consists of two subiterations of EM, after which the lexicon is pruned. Pruning is based on Viterbi counts (EM+Viterbi-prune). First, subwords that do not occur in the Viterbi segmentation are pre-pruned. The cost function is the estimated change in likelihood when the subword is removed, estimated using the assumption that all probability mass of the removed subword goes to its Viterbi segmentation. Subwords are sorted according to the cost, and a fixed proportion of remaining subwords are pruned each iteration. Single character

[2] https://github.com/google/sentencepiece

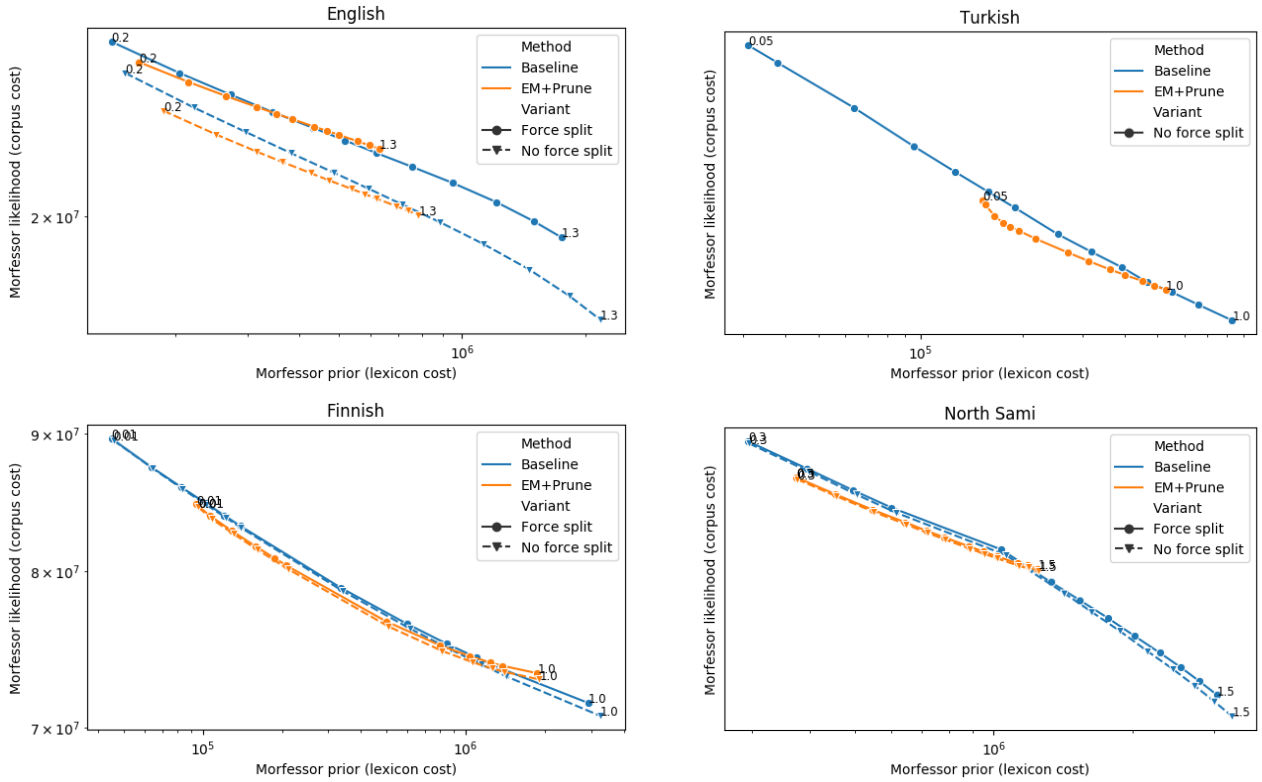


Figure 1: Unweighted Morfessor cost function components (prior and likelihood). Log scale.

subwords are never pruned. A predetermined lexicon size is used as the stopping condition.

3. Morfessor EM+Prune

Morfessor EM+Prune[3] uses the unigram language model and priors similar to Morfessor Baseline, but combines them with EM+Prune training.

3.1. Prior

The prior must be slightly modified for use with the EM+Prune algorithm. The prior for the frequency distribution (5) is derived using combinatorics. When using real-valued expected counts, there are infinite assignments of counts to parameters. Despite not being theoretically motivated, it can still be desirable to compute an approximation of the Baseline frequency distribution prior, in order to use EM+Prune as an improved search to find more optimal parameters for the original cost. To do this, the real valued token count ν is rounded to the nearest integer[4]. Alternatively, the prior for the frequency distribution can be omitted, or a new prior with suitable properties could be formulated. We do not propose a completely new prior in this work, instead opting to remain as close as possible to Morfessor Baseline.

[3]Software available at `https://github.com/Waino/morfessor-emprune` .

[4]An alternative would be to replace the factorial with the gamma function. This added precision serves no practical purpose, particularly as we already use Stirling's approximation of the factorial.

In Morfessor EM+Prune, morphs are explicitly stored in the lexicon, and morphs are removed from the lexicon only during pruning. This differs from Morfessor Baseline, in which a morph is implicitly considered to be stored in the lexicon if it has non-zero count.

The prior for the morph form properties does not need to be modified. During the EM parameter estimation, the prior for the morph form properties is omitted as the morph lexicon remains constant. During pruning, the standard form prior is applicable.

Additionally we apply the Bayesian EM implicit Dirichlet Process prior (Liang and Klein, 2007). We experiment with four variations of the prior:

1. the full EM+Prune prior,
2. omitting the Bayesian EM (noexpΨ),
3. omitting the approximate frequency distribution prior (nofreqdistr),
4. and omitting the prior entirely (noprior).

3.2. Seed Lexicon

The seed lexicon consists of the one million most frequent substrings, with two restrictions on which substrings to include: pre-pruning of redundant subwords, and forcesplit. Truncating to the chosen size is performed after pre-pruning, which means that pre-pruning can make space for substrings that would otherwise have been below the threshold.

Pre-pruning of ***redundant subwords*** is based on occurrence counts. If a string x occurs n times, then any substring of x will occur at least n times. Therefore, if

	FS	Prior ↓	Likelihood ↓	W-sum ↓
Words		1.79×10^{7}	1.32×10^{7}	2.98×10^{7}
Characters		2.35×10^{2}	2.90×10^{7}	2.61×10^{7}
EM+P MDL	✓	$\mathbf{4.69\times10^{5}}$	2.09×10^{7}	1.92×10^{7}
Morfessor Baseline	✓	7.55×10^{5}	2.05×10^{7}	1.92×10^{7}
Morfessor Baseline		8.84×10^{5}	$\mathbf{1.99\times10^{7}}$	$\mathbf{1.88\times10^{7}}$
EM+P MDL		5.80×10^{5}	2.02×10^{7}	$\mathbf{1.88\times10^{7}}$
EM+P MDL lateen		6.35×10^{5}	2.01×10^{7}	$\mathbf{1.88\times10^{7}}$

Table 2: Morfessor cost results for English. $\alpha = 0.9$. FS is short for forcesplit, W-sum for weighted sum of prior and likelihood. ↓means that lower values are better. The bolded method is our primary configuration.

	FS	Prior ↓	Likelihood ↓	W-sum ↓
Words		8.64×10^{7}	4.77×10^{7}	8.74×10^{7}
Characters		2.46×10^{2}	1.27×10^{8}	2.54×10^{6}
Morfessor Baseline	✓	$\mathbf{8.31\times10^{4}}$	8.60×10^{7}	1.80×10^{6}
Morfessor Baseline		8.36×10^{4}	8.59×10^{7}	1.80×10^{6}
EM+P MDL	✓	1.29×10^{5}	8.28×10^{7}	1.79×10^{6}
EM+P MDL lateen		1.41×10^{5}	$\mathbf{8.22\times10^{7}}$	1.79×10^{6}
EM+P MDL		1.31×10^{5}	8.26×10^{7}	$\mathbf{1.78\times10^{6}}$

Table 3: Morfessor cost results for Finnish. $\alpha = 0.02$.

the substring has a count of exactly n, we know that it is not needed in any other context except as a part of x. Such unproductive substrings are likely to be poor candidate subwords, and can be removed to make space in the seed lexicon for more useful substrings. This pre-pruning is not a neutral optimization, but does affect segmentation results. We check all initial and final substrings for redundancy, but do not pre-prune internal substrings.

To achieve ***forced splitting*** before or after certain characters, e.g. hyphens, apostrophes and colons, substrings which include a forced split point can be removed from the seed lexicon. As EM+Prune is unable to introduce new subwords, this pre-pruning is sufficient to guarantee the forced splits. While Morfessor 2.0 only implements force splitting certain characters to single-character morphs, i.e. force splitting on both sides, we implement more fine-grained force splitting separately before and after the specified character.

3.3. Training Algorithm

We experiment with three variants of the EM+Prune iteration structure:

1. EM,
2. Lateen-EM,
3. EM+Viterbi-prune

EM+Viterbi-prune is an intermediary mode between EM and lateen-EM in the context of pruning. The pruning decisions are made based on counts from a single iteration of Viterbi training, but these Viterbi counts are not otherwise used to update the parameters. In effect, this allows for the more aggressive pruning using the Viterbi counts, while retaining the uncertainty of the soft parameters.

Each iteration begins with 3 sub-iterations of EM. In the pruning phase of each iteration, the subwords in the current lexicon are sorted in ascending order according to the estimated change in the cost function if the subword is removed from the lexicon. Subwords consisting of a single character are always kept, to retain the ability to represent an open vocabulary without OOV issues. The list is then pruned according to one of three available pruning criteria:[5]

1. (α-weighted) MDL pruning,
2. MDL with automatic tuning of α for lexicon size,
3. lexicon size with omitted prior or pretuned α.

In (α-weighted) Minimum Description Length (MDL) pruning, subwords are pruned until the estimated cost starts rising, or until the pruning quota for the iteration is reached, whichever comes first.

A subword lexicon of a predetermined size can be used as pruning criterion in two different ways. If the desired α is known in advance, or if the prior is omitted, subwords are pruned until the desired lexicon size is reached, or until the pruning quota for the iteration is reached, whichever comes first.

To reach a subword lexicon of a predetermined size while using the Morfessor prior, the new ***automatic tuning*** procedure can be applied. For each subword, the estimated change in prior and likelihood are computed separately. These allow computing the value of α that would cause the removal of each subword to be cost neutral, i.e. the value that would cause MDL pruning to terminate at that subword. For subwords with the same sign for both the change in prior and likelihood, no such threshold α can be computed: if the removal decreases both costs the subword will always be removed, and if it increases both costs it will always be kept. Sorting the list of subwords according to the estimated threshold α including the always kept subwords allows automatically tuning α so that a sub-

	Prior ↓	Likelihood ↓	W-sum ↓
Words	1.31×10^{7}	9.09×10^{6}	1.68×10^{7}
Characters	1.19×10^{2}	2.08×10^{7}	8.30×10^{6}
Morfessor Baseline	$\mathbf{2.54\times10^{5}}$	1.39×10^{7}	5.82×10^{6}
EM+P MDL lateen	2.79×10^{5}	1.37×10^{7}	5.78×10^{6}
EM+P MDL	2.71×10^{5}	1.37×10^{7}	5.77×10^{6}
EM+P MDL keep-redundant	2.97×10^{5}	$\mathbf{1.36\times10^{7}}$	$\mathbf{5.73\times10^{6}}$

Table 4: Morfessor cost results for Turkish. $\alpha = 0.4$

	FS	Prior ↓	Likelihood ↓	W-sum ↓
Words		2.12×10^{7}	1.03×10^{7}	3.15×10^{7}
Characters		1.38×10^{3}	2.98×10^{7}	2.98×10^{7}
Morfessor Baseline	✓	1.76×10^{6}	1.62×10^{7}	1.80×10^{7}
Morfessor Baseline		1.87×10^{6}	$\mathbf{1.61\times10^{7}}$	1.80×10^{7}
EM+P MDL	✓	9.52×10^{5}	1.70×10^{7}	$\mathbf{1.79\times10^{7}}$
EM+P MDL lateen		9.83×10^{5}	1.69×10^{7}	$\mathbf{1.79\times10^{7}}$
EM+P MDL		9.56×10^{5}	1.69×10^{7}	$\mathbf{1.79\times10^{7}}$

Table 5: Morfessor cost results for North Sámi. $\alpha = 1.0$

[5]MDL with or without automatic tuning is not compatible with omitting the prior.

	α	FS	Pre ↑	Rec ↑	F ↑	
EM+P MDL noexpΨ	0.8	✓	82.9	71.8	**77.0**	
EM+P MDL nofreqdistr	0.8	✓	83.3	71.4	76.9	
EM+P MDL	0.9	✓	81.9	72.1	76.7	–
Morfessor Baseline	0.8	✓	**85.0**	68.5	75.9	–
Morfessor Baseline	0.7		83.8	69.4	75.9	∼**B**
EM+P MDL	0.6		79.0	**72.8**	75.8	
SentencePiece 50k	–		75.9	61.9	68.2	

Table 6: Boundary Precision (Pre), Recall (Rec) and F_1-score (F) results for English. ∼**E** indicates **not** significantly different (two-sided Wilcoxon signed-rank test, $p < 0.05$, zero splitting) from the bolded EM+Prune method, and ∼**B** from the bolded Baseline.

	α	FS	Pre ↑	Rec ↑	F ↑	
EM+P MDL	0.035	✓	72.0	55.8	**62.9**	–
EM+P MDL nofreqdistr	0.02	✓	68.7	58.0	**62.9**	∼**E**
EM+P MDL noexpΨ	0.02	✓	68.4	57.9	62.7	∼**E**
EM+P MDL	0.015		66.7	**58.5**	62.3	∼**E**
Morfessor Baseline	0.02	✓	62.3	58.2	60.2	–
SentencePiece 50k	–		**75.7**	49.3	59.7	∼**B**
Morfessor Baseline	0.02		62.0	57.6	59.7	

Table 7: Boundary Precision (Pre), Recall (Rec) and F_1-score (F) results for Finnish.

word lexicon of exactly the desired size is retained after MDL pruning. The automatic tuning is repeated before the pruning phase of each iteration, as retraining the parameters affects the estimates.

3.4. Sampling of Segmentations

Morfessor EM+Prune can be used in subword regularization (Kudo, 2018), a denoising-based regularization method for neural NLP systems. Alternative segmentations can be sampled from the full data distribution using Forward-filtering backward-sampling algorithm (Scott, 2002) or approximatively but more efficiently from an n-best list.

3.5. SentencePiece as a Special Case of Morfessor EM+Prune

Table 1 contains a comparison between all four methods discussed in this work. To recover SentencePiece, Morfessor EM+Prune should be run with the following settings: The prior should be omitted entirely, leaving only the likelihood

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \{-\log P(\boldsymbol{D} \mid \boldsymbol{\theta})\} \tag{7}$$

As the tuning parameter α is no longer needed when the prior is omitted, the pruning criterion can be set to a predetermined lexicon size, without automatic tuning of α. Morfessor by default uses type-based training; to use frequency information, count dampening should be turned off. The seed lexicon should be constructed without using forced splitting. The EM+Viterbi-prune training scheme should be used, with Bayesian EM turned on.

	α	Pre ↑	Rec ↑	F ↑	
EM+P MDL keep-redundant	0.3	**87.8**	58.7	**70.4**	
EM+P MDL noexpΨ	0.4	87.6	58.1	69.9	
EM+P MDL nofreqdistr	0.3	86.4	58.2	69.6	∼**E**
EM+P MDL	0.2	84.8	58.7	69.4	–
Morfessor Baseline	0.2	78.2	58.4	66.9	–
SentencePiece 12k	–	75.2	**60.0**	66.8	∼**B**

Table 8: Boundary Precision (Pre), Recall (Rec) and F_1-score (F) results for Turkish.

	α	FS	Pre ↑	Rec ↑	F ↑	
Morfessor Baseline	1.4		**75.7**	60.7	**67.4**	∼**E** –
EM+P MDL nofreqdistr	1.0	✓	73.7	61.8	67.2	∼**B**
Morfessor Baseline	1.2	✓	75.7	60.4	67.2	∼**E**∼**B**
EM+P MDL	1.3	✓	73.0	62.1	67.1	– ∼**B**
EM+P MDL	1.2		72.8	62.0	66.9	
EM+P MDL noexpΨ	0.4	✓	66.5	**65.9**	66.2	
SentencePiece 64k	–		65.3	61.3	63.3	

Table 9: Boundary Precision (Pre), Recall (Rec) and F_1-score (F) results for North Sámi.

4. Experimental Setup

English, Finnish and Turkish data are from the *Morpho Challenge 2010* data set (Kurimo et al., 2010a; Kurimo et al., 2010b). The training sets contain ca 878k, 2.9M and 617k word types, respectively. As test sets we use the union of the 10 official test set samples. For North Sámi, we use a list of ca 691k word types extracted from *Den samiske tekstbanken* corpus (Sametinget, 2004). and the 796 word type test set from version 2 of the data set collected by (Grönroos et al., 2015; Grönroos et al., 2016).

In most experiments we use a grid search with a development set to find a suitable value for α. The exception is experiments using autotuning or lexicon size criterion, and experiments using SentencePiece. We use type-based training (dampening counts to 1) with all Morfessor methods.

For English, we force splits before and after hyphens, and before apostrophes, e.g. *"women's-rights"* is force split into *"women +'s +- +rights"*. For Finnish, we force splits before and after hyphens, and after colons. For North Sámi, we force splits before and after colons. For Turkish, the Morpho Challenge data is preprocessed in a way that makes force splitting ineffectual.

4.1. Evaluation

The ability of the training algorithm to find parameters minimizing the Morfessor cost is evaluated by using the trained model to segment the training data, and loading the resulting segmentation as if it was a Morfessor Baseline model. We observe both unweighted prior and likelihood, and their α-weighted sum.

The closeness to linguistic segmentation is evaluated by comparison with annotated morph boundaries using *boundary precision*, *boundary recall*, and *boundary F_1-score* (Virpioja et al., 2011). The boundary F_1-score (F-score for short) equals the harmonic mean of precision (the percentage of correctly assigned boundaries with respect to all assigned boundaries) and recall

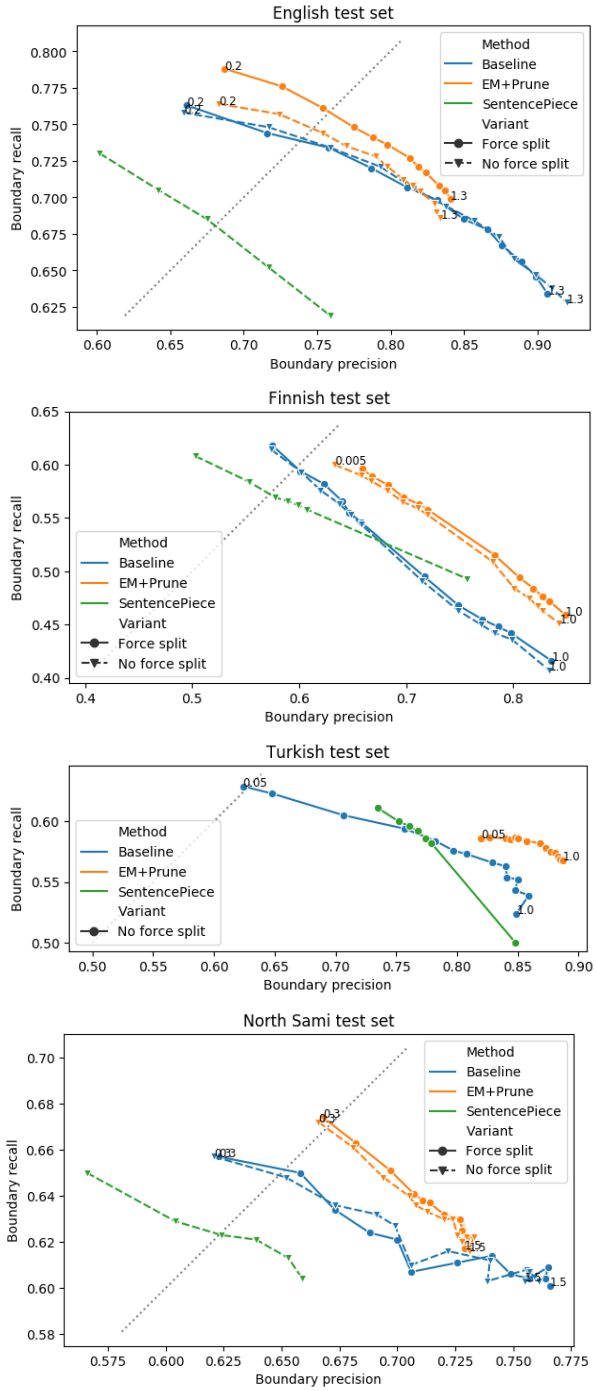


Figure 2: Boundary Precision–Recall curve at different tuning points, The smallest and largest α-values are labeled.

(the percentage of correctly assigned boundaries with respect to the reference boundaries). Precision and recall are calculated using macro-averages over the word types in the test set. In the case that a word has more than one annotated segmentation, we take the one that gives the highest score.

4.2. Error Analysis

We perform an error analysis, with the purpose of gaining more insight into the ability of the methods to model particular aspects of morphology. We follow the procedure used by Ruokolainen et al. (2016). It is based on a categorization of morphs into the categories PREFIX, STEM, and SUFFIX. The category labels are derived from the original morphological analysis labels in the English and Finnish gold standards, and directly correspond to the annotation scheme used in the North Sámi test set.

We first divide errors into two kinds, *over-segmentation* and *under-segmentation*. Over-segmentation occurs when a boundary is incorrectly assigned within a morph segment. In under-segmentation, the a correct morph boundary is omitted from the generated segmentation. We further divide the errors by the morph category in which the over-segmentation occurs, and the two morph categories surrounding the omitted boundary in under-segmentation.

5. Results

Figure 1 compares the cost components of the Morfessor model across different α parameters. The lowest costs for the mid-range settings are obtained for the EM+Prune algorithm, but for larger lexicons, the Baseline algorithm copes better. As expected, using forced splits at certain characters increase the costs, and the increase is larger than between the training algorithms. As Turkish preprocessing causes the results to be unaffected by the forced splits, we only report results without them.

Tables 2 to 5 show the Morfessor cost of the segmented training data for particular α values. Again, the proposed Morfessor EM+Prune reaches a lower Morfessor cost than Morfessor Baseline. Using the lateen-EM has only minimal effect to the costs, decreasing the total cost slightly for English and increasing for the other languages. Turkish results include the "keep-redundant" setting discussed below in more detail.

Figure 2 shows the Precision–Recall curves for the primary systems, for all four languages. While increasing the Morfessor cost, forced splitting improves BPR. Tables 6 to 9 show test set Boundary Precision, Recall and F_1-score (BPR) results at the optimal tuning point (selected using a development set) for each model, for English, Finnish, Turkish and North Sámi, respectively[6]. The default Morfessor EM+Prune configuration ("soft" EM, full prior, forcesplit) significantly outperforms Morfessor Baseline w.r.t. the F-score for all languages except North Sámi, for which there is no significant difference between the methods.

Morfessor EM+Prune is less responsive to tuning than Morfessor Baseline. This is visible in the shorter lines in Figures 1 and 2, although the tuning parameter takes values from the same range. In particular, EM+Prune can not easily be tuned to produce very large lexicons.

[6]Note that SentencePiece is not designed for aiming towards a linguistic morpheme segmentation. Neither does it attempt to minimize the Morfessor cost. Therefore, SentencePiece is included in the evaluations for context, not as a baseline method.

		Over-segmentation					Under-segmentation						
Language	Model	STM ↓	SUF ↓	PRE ↓	UNKNOWN ↓	**PRE/TOT** ↑	STM-SUF ↓	STM-STM ↓	SUF-SUF ↓	SUF-STM ↓	PRE-STM ↓	UNKNOWN ↓	**REC/TOT** ↑
eng	Characters	71.05	11.82	1.66	0.33	15.13	0.00	0.00	0.00	0.00	0.00		100.00
eng	Words	0.00	0.00	0.00	0.00	100.00	55.07	5.90	8.56	0.14	4.38		23.06
eng	SentencePiece 38k	17.60	10.25	0.18	0.24	71.74	26.40	2.48	2.74	0.05	2.78		65.26
eng	Morfessor Baseline	**10.17**	**2.32**	**0.03**	**0.07**	**87.42**	22.46	2.10	4.75	**0.04**	1.65		67.37
eng	EM+Prune MDL	15.46	2.75	0.05	0.13	81.61	**19.93**	**1.82**	**4.32**	**0.04**	**1.46**		**70.84**
fin	Characters	65.23	13.80	0.67	0.57	19.73	0.00	0.00	0.00	0.00	0.00	0.00	100.00
fin	Words	0.00	0.00	0.00	0.00	100.00	49.19	17.16	21.76	4.84	0.96	0.58	4.09
fin	SentencePiece 13k	35.11	3.71	0.08	0.41	60.69	25.96	1.45	16.18	0.35	0.08	**0.16**	**55.81**
fin	Morfessor Baseline	34.75	2.82	**0.03**	0.38	62.02	**24.57**	**0.86**	16.31	**0.15**	**0.04**	0.20	57.63
fin	EM+Prune MDL	**29.34**	**2.20**	**0.03**	**0.26**	**68.18**	24.68	0.90	**15.95**	0.29	0.05	0.19	57.60
sme	Characters	81.44	6.80			11.76	0.00	0.00	0.00	0.00			100.00
sme	Words	0.00	0.00			100.00	52.92	13.15	4.43	0.61			28.64
sme	SentencePiece 64k	30.10	4.52			65.38	31.35	3.96	**3.09**	0.20			61.40
sme	Morfessor Baseline	**23.27**	**3.02**			**73.71**	33.16	**2.22**	3.40	**0.10**			60.99
sme	EM+Prune MDL	23.35	4.41			72.25	**30.48**	3.10	3.23	0.17			**62.84**

Table 10: Error analysis for English (eng, $\alpha = 0.9$), Finnish (fin, $\alpha = 0.02$), and North Sámi (sme, $\alpha = 1.0$). All results without forcesplit. Over-segmentation and under-segmentation errors reduce precision and recall, respectively.

Pre-pruning of redundant substrings gives mixed results. For Turkish, both Morfessor cost and BPR are degraded by the pre-pruning, but for the other three languages the pre-pruning is beneficial or neutral. When tuning α to very high values (less segmentation), pre-pruning of redundant substrings improves the sensitivity to tuning. The same effect may also be achievable by using a larger seed lexicon. We perform most of our experiments with pre-pruning turned on.

To see the effect of pre-pruning on the seed lexicon, we count the number of subwords that are used in the gold standard segmentations, but not included in seed lexicons of various sizes. Taking Finnish as an example, we see 203 subword types missing from a 1 million substring seed lexicon without pre-pruning. Turning on pre-pruning decreases the number of missing types to 120. To reach the same number without using pre-pruning, a much larger seed lexicon of 1.7M substrings must be used.

Omitting the frequency distribution appears to have little effect on Morfessor cost and BPR. Turning off Bayesian EM (noexpΨ) results in a less compact lexicon resulting in higher prior cost, but improves BPR for two languages: English and Turkish.

Table 10 contains the error analysis for English, Finnish and North Sámi. For English and North Sámi, EM+Prune results in less under-segmentation but worse over-segmentation. For Finnish these results are reversed. However, the suffixes are often better modeled, as shown by lower under-segmentation on SUF-SUF (all languages) and STM-SUF (English and North Sámi).

6. Conclusion

We propose Morfessor EM+Prune, a new training algorithm for Morfessor Baseline. EM+Prune reduces search error during training, resulting in models with lower Morfessor costs. Lower costs also lead to improved accuracy when segmentation output is compared to linguistic morphological segmentation.

We compare Morfessor EM+Prune to three previously published segmentation methods applying unigram language models. We find that using the Morfessor prior is beneficial when the reference is linguistic morphological segmentation.

In this work we focused on model cost and linguistic segmentation. In future work the performance of Morfessor EM+Prune in applications will be evaluated. Also, a new frequency distribution prior, which is theoretically better motivated or has desirable properties, could be formulated.

7. Acknowledgements

This study has been supported by the MeMAD project, funded by the European Union's Horizon 2020 research and innovation programme (grant agreement № 780069), and the FoTran project, funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement № 771113) Computer resources within the Aalto University School of Science "Science-IT" project were used.

8. Bibliographical References

Arivazhagan, N., Bapna, A., Firat, O., Lepikhin, D., Johnson, M., Krikun, M., Chen, M. X., Cao, Y., Foster, G., Cherry, C., Macherey, W., Chen, Z., and Wu, Y. (2019). Massively multilingual neural machine translation in the wild: Findings and challenges. *CoRR*, abs/1907.05019.

Cotterell, R., Kirov, C., Sylak-Glassman, J., Walther, G., Vylomova, E., Xia, P., Faruqui, M., Kübler, S., Yarowsky, D., Eisner, J., and Hulden, M. (2017). CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30, Vancouver, August. Association for Computational Linguistics.

Creutz, M. and Lagus, K. (2002). Unsupervised discovery of morphemes. In *ACL-02 Workshop on Morphological and Phonological Learning*, volume 6 of *MPL '02*, pages 21–30, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Creutz, M. and Lagus, K. (2007). Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1), January.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38.

Gage, P. (1994). A new algorithm for data compression. *C Users Journal*, 12(2):23–38, February.

Grönroos, S.-A., Hiovain, K., Smit, P., Rauhala, I. E., Jokinen, P. K., Kurimo, M., and Virpioja, S. (2016). Low-resource active learning of morphological segmentation. *Northern European Journal of Language Technology*.

Grönroos, S.-A., Jokinen, K., Hiovain, K., Kurimo, M., and Virpioja, S. (2015). Low-resource active learning of North Sámi morphological segmentation. In *Proceedings of 1st International Workshop in Computational Linguistics for Uralic Languages*, pages 20–33.

Hammarström, H. and Borin, L. (2011). Unsupervised learning of morphology. *Computational Linguistics*, 37(2):309–350, June.

Kohonen, O., Virpioja, S., and Lagus, K. (2010). Semi-supervised learning of concatenative morphology. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 78–86, Uppsala, Sweden, July. Association for Computational Linguistics.

Kudo, T. and Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November. Association for Computational Linguistics.

Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv:1804.10959 [cs]*, April. arXiv: 1804.10959.

Kurimo, M., Virpioja, S., Turunen, V., and Lagus, K. (2010a). Morpho challenge 2005-2010: Evaluations and results. In Jeffrey Heinz, et al., editors, *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 87–95, Uppsala, Sweden, July. Association for Computational Linguistics.

Kurimo, M., Virpioja, S., and Turunen, V. T. (2010b). Overview and results of Morpho Challenge 2010. In *Proceedings of the Morpho Challenge 2010 Workshop*, pages 7–24, Espoo, Finland, September. Aalto University School of Science and Technology, Department of Information and Computer Science.

Liang, P. and Klein, D. (2007). Structured Bayesian nonparametric models with variational inference (tutorial). In *Association for Computational Linguistics (ACL)*.

Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry*, volume 15. World Scientific Series in Computer Science, Singapore.

Ruokolainen, T., Kohonen, O., Sirts, K., Grönroos, S.-A., Kurimo, M., and Virpioja, S. (2016). A comparative study on minimally supervised morphological segmentation. *Computational Linguistics*.

Scott, S. L. (2002). Bayesian methods for hidden markov models: Recursive computing in the 21st century. *Journal of the American Statistical Association*, 97(457):337–351.

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. In *ACL16*, August. arXiv: 1508.07909.

Smit, P., Virpioja, S., Kurimo, M., et al. (2017). Improved subword modeling for WFST-based speech recognition. In *In INTERSPEECH 2017–18th Annual Conference of the International Speech Communication Association*.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2011). Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1269–1280. Association for Computational Linguistics.

Varjokallio, M., Kurimo, M., and Virpioja, S. (2013). Learning a subword vocabulary based on unigram likelihood. In *Proc. 2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 7–12, Olomouc, Czech Republic, December. IEEE.

Virpioja, S., Turunen, V. T., Spiegler, S., Kohonen, O., and Kurimo, M. (2011). Empirical comparison of evaluation methods for unsupervised learning of morphology. *Traitement Automatique des Langues*, 52(2):45–90.

Virpioja, S., Smit, P., Grönroos, S.-A., and Kurimo, M. (2013). Morfessor 2.0: Python implementation and

extensions for morfessor baseline. Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, Department of Signal Processing and Acoustics, Aalto University, Helsinki, Finland.

Virpioja, S. (2012). *Learning Constructions of Natural Language: Statistical Models and Evaluations*. PhD Thesis, Aalto University, Espoo, Finland, December.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

9. Language Resource References

Grönroos, Stig-Arne and Hiovain, Katri and Smit, Peter and Rauhala, Ilona Erika and Jokinen, Päivi Kristiina and Kurimo, Mikko and Virpioja, Sami. (2015). *North Sámi active learning morphological segmentation annotations*. Aalto University, 2.0.

Kudo, Taku and Richardson, John. (2018). *SentencePiece*. Taku Kudo.

Kurimo, Mikko and Virpioja, Sami and Turunen, Ville T. (2010). *Morpho Challenge 2010 dataset*. Aalto University.

Sametinget. (2004). *Den samiske tekstbanken*. UiT Norgga árktalaš universitehta.

Virpioja, Sami and Smit, Peter and Grönroos, Stig-Arne and Kurimo, Mikko. (2013). *Morfessor 2.0*. Aalto University, 2.0.6.

Publication VI

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. Hybrid morphological segmentation for phrase-based machine translation. In *Proceedings of the First Conference on Machine Translation*, Berlin, Germany, pages 289–295, Aug 2016.

Hybrid Morphological Segmentation for Phrase-Based Machine Translation

Stig-Arne Grönroos
Department of Signal Processing and Acoustics
Aalto University, Finland
stig-arne.gronroos@aalto.fi

Sami Virpioja
Department of Computer Science
Aalto University, Finland
sami.virpioja@aalto.fi

Mikko Kurimo
Department of Signal Processing and Acoustics
Aalto University, Finland
mikko.kurimo@aalto.fi

Abstract

This article describes the Aalto University entry to the English-to-Finnish news translation shared task in WMT 2016. Our segmentation method combines the strengths of rule-based and unsupervised morphology. We also attempt to correct errors in the boundary markings by post-processing with a neural morph boundary predictor.

1 Introduction

Using words as translation tokens is problematic for synthetic languages with rich inflection, derivation or compounding. Such languages have very large vocabularies, leading to sparse statistics and many out-of-vocabulary words. Differences in morphological complexity between source and target languages also complicate alignment.

A common method for alleviating these problems is to segment the morphologically richer side as a pre-processing step. Over-segmentation is detrimental, however, as longer windows of history need to be used, and useful phrases become more difficult to extract. It is therefore important to find a balance in the amount of segmentation.

We consider the case that there are linguistic gold standard segmentations available for the morphologically complex target language. Even if there is no rule-based morphological analyzer for the language, a limited set of gold standard segmentations can be used for training a reasonably accurate statistical segmentation model in a supervised or semi-supervised manner (Ruokolainen et al., 2014; Cotterell et al., 2015).

While using a linguistically accurate morphological segmentation in a phrase-based SMT system may sound like a good idea, there is evidence that shows otherwise. In general, over-segmentation seems to be a larger problem for NLP applications than under-segmentation (Virpioja et al., 2011). In the case of SMT, linguistic morphs may provide too high granularity compared to the second language, and deteriorate alignment (Habash and Sadat, 2006; Chung and Gildea, 2009; Clifton and Sarkar, 2011). Moreover, longer sequences of units are needed in the language model and the translation phrases to cover the same span of text.

An unsupervised morphological segmentation may alleviate these problems. A method based on optimizing the training data likelihood, such as Morfessor (Creutz and Lagus, 2002; Creutz and Lagus, 2007; Virpioja et al., 2013), ensures that common phenomena are modeled more accurately, for example by using full forms for highly-frequent words even if they consist of multiple morphemes. Data-driven methods also allow tuning the segmentation granularity, for example based on symmetry between the languages in a parallel corpus (Grönroos et al., 2015).

To combine the advantages of linguistic segmentation and data-driven segmentation, we propose a hybrid approach for morphological segmentation. We optimize the segmentation in a data-driven manner, aiming for a similar granularity as the second language of the language pair, but restricting the possible set of segmentation boundaries to those between linguistic morphs. That is, the segmentation method may decide to join any of the linguistic morphs, but it cannot add new segmentation boundaries to known linguistic morphs.

We show that it is possible to improve on the linguistically accurate segmentation by reducing the amount of segmentation in an unsupervised manner.

1.1 Related work

Rule-based and statistical segmentation for SMT have been extensively studied in isolation (Virpi-

Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers, pages 289–295,
Berlin, Germany, August 11-12, 2016.

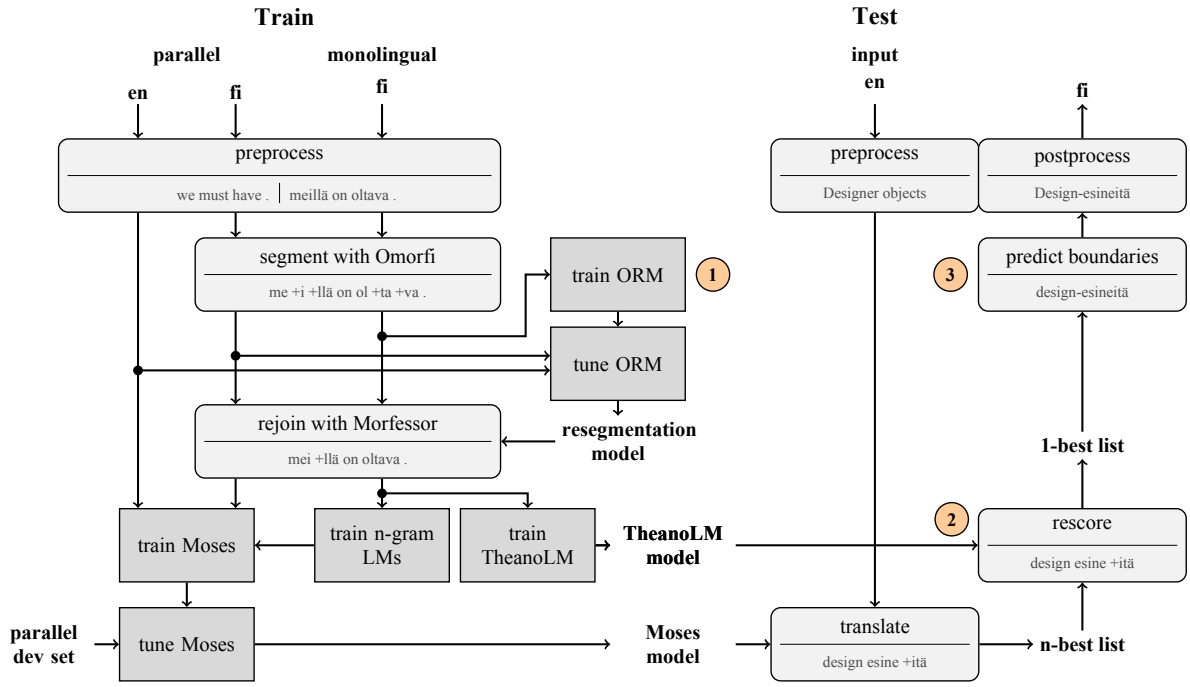


Figure 1: A pipeline overview of training of the system and using it for translation. Main contributions are hilighted with numbers 1-3. ORM is short for Omorfi-restricted Morfessor.

oja et al., 2007; Fishel and Kirik, 2010; Luong et al., 2010), and also the use of system combination to combine their strengths has been examined (De Gispert et al., 2009; Rubino et al., 2015; Pirinen et al., 2016).

Prediction of morph boundary types has been used in conjunction with compound splitting. Stymne and Cancedda (2011) apply rule-based compound splitting in the pre-processing stage, and a conditional random field with rich linguistic features for generating novel compounds in post-processing. Coalescence of compound parts in the translation output is promoted using POS-tag features. Cap et al. (2014) extend the post-predictor to also inflect the compound modifiers e.g. to add a linking morpheme.

Stymne et al. (2013) investigate several methods for splitting and merging compounds when translating into Germanic languages, and provide an extensive reading list on the topic.

2 System overview

An overview of the system is shown in Figure 1. The three main contributions of this work are indicated by numbered circles:

1. Combining rule-based morphological segmentation (Omorfi) to data-driven morphological segmentation (Morfessor).
2. Rescoring n-best lists with TheanoLM (Enarvi and Kurimo, 2016).
3. Correcting boundary markings with post-processing predictor.

Our system extends the phrase-based SMT system Moses (Koehn et al., 2007) to perform segmented translation, by adding pre-processing and post-processing steps, with no changes to the decoder.

The standard pre-processing steps not specified in Figure 1 consist of normalization of punctuation, tokenization, and statistical truecasing. All of these were performed with the tools included in Moses. The pre-processing steps are followed by morphological segmentation.

In addition, the parallel data was cleaned and duplicate sentences were removed. Cleaning was performed after morphological segmentation, as the segmentation can increase the length in tokens of a sentence.

The post-processing steps include rescoring of the n-best list, boundary prediction and desegmentation. These are followed by the standard post-processing steps, reversing the pre-processing steps: detruecasing and detokenization.

System	Tokens	Segmentation		
Words	3	hyötyajoneuvojen	tekniset	tienvarsitarkastukset
		[commercial vehicles']	*[technical]*	*[roadside inspections]*
Omorfi	11	hyöty@ ajo@ neuvo +j +en	teknise +t	tien@ varsi@ tarkastukse +t
		[utility] [drive] [counsel] [+Pl] [+Gen]	*[technical] [+Pl]*	*[road] [side] [inspection] [+Pl]*
ORM	5	hyötyajoneuvo +jen	tekniset	tienvarsi@ tarkastukset
		[commercial vehicle] [+Pl +Gen]	*[technical]*	*[roadside] [inspections]*
Source	6	technical roadside inspection of commercial vehicles		

Table 1: Worked example of two-stage morphological segmentation, beginning with rule-based Omorfi segmentation and followed by Omorfi-restricted Morfessor (ORM). The glosses below the segmentations show approximate meaning of the segments (Pl = plural suffix, Gen = genitive suffix).

2.1 Morphological segmentation

An example of the morphological segmentation is shown in Table 1.

2.1.1 Omorfi segmentation

We begin the morphological segmentation by applying the segmentation tool from Omorfi (Pirinen, 2015). Hyphens removed by Omorfi are reintroduced.

Omorfi outputs 5 types of intra-word boundaries, which we mark in different ways. Compound modifiers, identified by the `WB` or `wB` boundary type, are marked with a reserved symbol '@' at the right edge of the morph. Suffixes, identified by a leading morph boundary `MB` or derivation boundary `DB`, are marked with a '+' at the left edge. Boundaries of the type `STUB` (other stemmer-type boundary) are removed. This marking scheme leaves the compound head, or last stem of the word, unmarked. E.g. "yli{WB}voimai{STUB}s{MB}i{MB}a" is marked as "yli@ voimais +i +a".

Words not identified by Omorfi are collected in a separate vocabulary, and treated as unsegmentable.

2.1.2 Restricted Morfessor Baseline

In order to force the Morfessor method to follow the linguistic morphs produced by Omorfi, we added some new features to the Morfessor Baseline implementation by Virpioja et al. (2013). The new extension, Restricted Morfessor Baseline, is able to remove any of the given intra-word boundaries, but cannot introduce any new ones.

The standard training algorithm of Morfessor iterates over the word forms, testing whether to split the corresponding string to two parts or leave it as it is. If the string is split, the testing descends recursively to the substrings. The segmentation decisions are stored in a binary tree structure, where each node corresponds to a string. The root nodes are full word forms and leaf nodes are morphs.

The middle nodes are substrings shared by several word forms, which means that if two word forms have different restrictions on the same substring, some of the restrictions may be violated. While the amount of violations was in practice very small, we ensured that no restrictions were violated in the end by applying the recursive algorithm only for the two first epochs, and then switching to Viterbi training.

In Viterbi training, each word is re-segmented to the most likely segmentation given the current model parameters using an extension of the Viterbi algorithm. We modified the implementation of Virpioja et al. (2013) to remove the previous segments of the word from the parameters before re-analyzing the word, and re-adding the segments of the new optimal segmentation afterwards. Additive smoothing with smoothing constant 1.0 was applied in the Viterbi search.

Prior to the Viterbi training, we flattened the tree structure so that the root nodes (word forms) link directly to the leaf nodes (morphs), thus removing any shared substrings nodes that are not actual morphs. This way all word forms are segmented independently and all the restrictions are followed.

2.1.3 Tuning the amount of segmentation

Omorfi-restricted Morfessor was tuned following Grönroos et al. (2015) to bring the number of tokens on the Finnish target side as close as possible to the English source side. The corpus weight hyper-parameter α was chosen by minimizing the sentence-level difference in token counts between the English and the segmented Finnish sides of the parallel corpus.

2.2 Rescoring n-best lists

Segmentation of the word forms increases the distances spanned by dependencies that should be modeled by the language model. To compensate for this, we apply a strong recurrent neural language model, TheanoLM. A recurrent language model is able to use arbitrarily long contexts without suffering from data sparsity, as opposed to n-gram language models, which are limited to a short context window. The additional language model is used in a separate rescoring step, to speed up translation, and for ease of implementation.

The TheanoLM model was trained on morphologically segmented data. Morphs occurring less than 1000 times in the full monolingual data were removed from the vocabulary, and replaced with the tag `<UNK>`. To create a class vocabulary, the morphs were embedded in a 300-dimensional space using word2vec (Mikolov et al., 2013). The embeddings were clustered into 2000 classes, using agglomerative clustering with cosine distance. Due to TheanoLM limitations, only the Europarl and News data (but not Common Crawl) were used for training.

The TheanoLM parameters were: 100 nodes in the projection layer, 300 LSTM nodes in the hidden layer, dropout rate 0.25, adam optimization with initial learning rate 0.01, and minibatch 16.

2.3 Morph boundary correction

One benefit of segmented translation is the ability to generate new compounds and inflections, that were not seen in the training data. However, the ability can also lead to errors, e.g when an English word frequently aligned to a compound modifier is translated using such a morph, even though there is no compound head to modify. The "dangling" morph boundary marker will then cause the space to be omitted, forming an incorrect compound with whatever word happens to follow.

For example, the Finnish pronoun *moni* (many) is also a frequent prefix, as in *monitoimi-* (multipurpose) or *monikulttuurinen* (multicultural). This resulted in an erroneous novel compound in *moniliberaalien keskuudessa* ("among the multiliberals"), which was corrected by introducing a space between *moni* and *liberaalien*, leading to a correct translation ("many among the liberals").

In the opposite type of error, compounds may be translated as separate words, or hyphenated compounds translated with the hyphen omitted.

We trained a neural network predictor to correct such errors by predicting the boundary type {space, empty, hyphen} as an additional post-processing step before joining the tokens.

The neural network takes as input both a token level representation, in the form of the same word2vec embeddings as used in rescoring, and a character level representation windowed to 4 characters before and after the boundary. The tokens are encoded by a bidirectional network of Gated Recurrent Units (Cho et al., 2014), while the characters are encoded by a feed-forward network.

Even though the boundary markers in the translation output are unreliable, they are a strong clue. Our predictor has access to the translated markers. During training markers were randomly corrupted to avoid relying too much on them.

2.4 Moses configuration

We used GIZA++ alignment. As decoding LMs, we used two SRILM n-gram models with modified-KN smoothing: a 3-gram and 5-gram model, trained from different data. Many Moses settings were left at their default values: phrase length 10, grow-diag-final-and alignment symmetrization, msd-bidirectional-fe reordering, and distortion limit 6.

The feature weights were tuned using MERT (Och, 2003), with BLEU (Papineni et al., 2002) of the post-processed hypothesis against a development set as the metric. 20 random restarts per MERT iteration were used, with iterations repeated until convergence.

The rescoring weights were tuned with a newly included script in Moses, which uses kb-MIRA instead of MERT.

3 Data

Our system participates in the constrained condition of the shared task. As parallel data, we used the Europarl-v8 and Wikititles corpora, resulting in 1 846 609 sentences after applying the Omorfi-restricted Morfessor segmentation and cleaning.

As monolingual data, we used the Finnish side of Europarl-v8, news.2014.fi.shuffled.v2, news.2015.fi.shuffled and Common Crawl. The total size of monolingual data after cleaning was 133 848 615 sentences, 2 135 919 860 morph tokens, and 11 771 367 morph types. Setting the frequency threshold to 1000 occurrences for the

Configuration	%BLEU, newstest 2015	2016	Example sentence Other applications could focus on muscle cells and insulin-producing cells, he added.
Omorfi-restricted Morfessor	10.77	11.27	Muissa sovelluksissa voi keskittyä lihas solujen ja insuliinia tuottavien solujen, hän lisäsi.
+boundary correction	10.83	11.27	Muissa sovelluksissa voi keskittyä lihassolujen ja insuliinia tuottavien solujen, hän lisäsi.
+rescoring	11.17	**11.73**	Muut sovellukset voivat keskittyä lihas soluja ja insuliinia tuottavia soluja, hän lisäsi.
+rescoring +boundary corr.	**11.21**	11.72	Muut sovellukset voivat keskittyä lihassoluja ja insuliinia tuottavia soluja, hän lisäsi.
Omorfi	10.00	10.59	Muut sovellukset voisi keskittyä lihassolujen ja insuliinia tuottavien soluja, hän lisäsi.
+boundary correction	10.07	10.61	Muut sovellukset voisi keskittyä lihassolujen ja insuliinia tuottavien soluja, hän lisäsi.
+rescoring	10.70	11.11	Muut sovellukset voivat keskittyä lihassoluja ja insuliinia tuottavien soluja, hän lisäsi.
+rescoring +boundary corr.	10.78	11.11	Muut sovellukset voivat keskittyä lihassoluja ja insuliinia tuottavien soluja, hän lisäsi.
Word baseline	10.48	10.65	Muut sovellukset voisivat keskittyä lihaksia ja insuliinia tuottavien solujen-, hän lisäsi.
Reference translation			Muut sovelluskohteet voisivat keskittyä lihassoluihin ja insuliinia tuottaviin soluihin, hän lisäsi.

Table 2: Results of automatic evaluation, in BLEU percentage points.

TheanoLM morph lexicon reduced the number of morph types to 121 735.

The complete monolingual data including the Common Crawl was only used for creating the morph lexicon and for training the 3-gram LM. For the 5-gram LM, the TheanoLM and the boundary predictor, the Common Crawl was omitted.

Because hyphenated compounds are much less frequent than non-hyphenated words, we enriched the training data for the boundary predictor by adding the list of words compounds containing a single hyphen and occurring more than 10 times in the full monolingual corpus.

4 Results

Results are summarized in Table 2, together with example translations produced by the different system configurations.

The Omorfi-restricted Morfessor segmentation leads consistently to an improvement over directly using the Omorfi segmentation. For all configurations on the newstest2016 set, and for newstest2015 without rescoring, the improvement is over +0.6 BLEU. On newstest2015 with rescoring, the improvement is slightly smaller, +0.47 BLEU.

Adding the TheanoLM rescoring increases BLEU between +0.4 and +0.7 BLEU. The increase is larger for the more aggressively segmented Omorfi system, supporting the conclusion that a strong language model is needed to compensate for the longer sequences.

In total, our best system results in a +1 BLEU improvement over the word baseline.

Boundary prediction gave a modest improvement of under +0.1 BLEU on the newstest2015 set, the effect on the newstest2016 set was neutral. While the predictor works reliably for the correct Finnish text it was trained on, manual inspection shows that the performance is erratic for disfluent translation output. Even while the minor cosmetic improvements are more common than errors, the benefit is hard to quantify.

Due to a mistake during data pre-processing, one of the n-gram language models penalizes the use of numbers. The problem affects all the evaluated systems and lowers the overall scores. However, it does not affect the increase in BLEU from the use of Omorfi-restricted Morfessor or rescoring. We verified this using BLEU of the test set with all source sentences containing numbers removed.

5 Conclusions

We propose a new morphological segmentation method, combining the strengths of rule-based and unsupervised morphology. We optimize the segmentation in a data-driven manner, aiming to balance granularity between the two languages, while restricting segmentation to a subset of the linguistic morph boundaries. Using this segmentation, we improve SMT quality over the linguistically accurate segmentation.

Using a neural morph boundary predictor to correct errors in the boundary markings does not lead to an improvement in BLEU.

In total, our best system results in a +1 BLEU improvement over the word baseline.

Acknowledgments

This research has been supported by the Academy of Finland under the Finnish Centre of Excellence Program 2012–2017 (grant n°251170), and LASTU Programme (grants n°256887 and 259934). Computer resources within the Aalto

University School of Science “Science-IT” project were used.

References

Fabienne Cap, Alexander M Fraser, Marion Weller, and Aoife Cahill. 2014. How to produce unseen teddy bears: Improved morphological processing of compounds in SMT. In *14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Gothenburg, Sweden. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar.

Tagyoung Chung and Daniel Gildea. 2009. Unsupervised tokenization for machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, Singapore. Association for Computational Linguistics.

Ann Clifton and Anoop Sarkar. 2011. Combining morpheme-based machine translation with post-processing morpheme prediction. In *ACL: HLT*, Portland, Oregon, USA, June. Association for Computational Linguistics.

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015. Labeled morphological segmentation with semi-markov models. In *CONLL*, Beijing, China. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *ACL-02 Workshop on Morphological and Phonological Learning*, Philadelphia, PA, USA, July. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1), January.

Adrià De Gispert, Sami Virpioja, Mikko Kurimo, and William Byrne. 2009. Minimum Bayes risk combination of translation hypotheses from alternative morphological decompositions. In *Proceedings of HLT-NAACL 2009: Short Papers*, Boulder, CO, USA. Association for Computational Linguistics.

Seppo Enarvi and Mikko Kurimo. 2016. TheanoLM - An Extensible Toolkit for Neural Network Language Modeling. *ArXiv e-prints*, May. `http://arxiv.org/abs/1605.00942`.

Mark Fishel and Harri Kirik. 2010. Linguistically motivated unsupervised segmentation for machine translation. In *Language Resources and Evaluation (LREC)*, Valletta, Malta.

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. 2015. Tuning phrase-based segmented translation for a morphologically complex target language. In *Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal, September. Association for Computational Linguistics.

Nizar Habash and Fatiha Sadat. 2006. Arabic preprocessing schemes for statistical machine translation. In *Proceedings of HLT-NAACL*, New York, USA. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *45th annual meeting of the ACL on interactive poster and demonstration sessions*, Prague, Czech Republic. Association for Computational Linguistics.

Minh-Thang Luong, Preslav Nakov, and Min-Yen Kan. 2010. A hybrid morpheme-word representation for machine translation of morphologically rich languages. In *Empirical Methods in Natural Language Processing (EMNLP)*, Cambridge, MA, USA. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems (NIPS)*, Lake Tahoe, NV, USA.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *41st Annual Meeting on Association for Computational Linguistics*, Sapporo, Japan. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *40th annual meeting of the association for computational linguistics*, Philadelphia, PA, USA. Association for Computational Linguistics.

Tommi A Pirinen, Antonio Toral, and Raphael Rubino. 2016. Rule-based and statistical morph segments in English-to-Finnish SMT. In *2nd International Workshop on Computational Linguistics for Uralic Languages*, Szeged, Hungary, Jan.

Tommi A Pirinen. 2015. Omorfi–Free and open source morphological lexical database for Finnish. In *20th Nordic Conference on Computational Linguistics (NODALIDA)*, Vilnius, Lithuania.

Raphael Rubino, Tommi Pirinen, Miquel Esplà-Gomis, Nikola Ljubešić, Sergio Ortiz Rojas, Vassilis Papavassiliou, Prokopis Prokopidis, and Antonio Toral. 2015. Abu-MaTran at WMT 2015 translation task: Morphological segmentation and web

crawling. In *Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal, September. Association for Computational Linguistics.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2014. Painless semi-supervised morphological segmentation using conditional random fields. In *European Chapter of the Association for Computational Linguistics (EACL)*, Gothenburg, Sweden, April. Association for Computational Linguistics.

Sara Stymne and Nicola Cancedda. 2011. Productive generation of compound words in statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Edinburgh, UK. Association for Computational Linguistics.

Sara Stymne, Nicola Cancedda, and Lars Ahrenberg. 2013. Generation of compound words in statistical machine translation into compounding languages. *Computational Linguistics*, 39(4).

Sami Virpioja, Jaakko J Väyrynen, Mathias Creutz, and Markus Sadeniemi. 2007. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. *Machine Translation Summit XI*, 2007.

Sami Virpioja, Ville T. Turunen, Sebastian Spiegler, Oskar Kohonen, and Mikko Kurimo. 2011. Empirical comparison of evaluation methods for unsupervised learning of morphology. *Traitement Automatique des Langues*, 52(2).

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. 2013. Morfessor 2.0: Python implementation and extensions for Morfessor Baseline. Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, Department of Signal Processing and Acoustics, Aalto University, Helsinki, Finland.

Publication VII

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. Tuning phrase-based segmented translation for a morphologically complex target language. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal, pages 105–111, Sep 2015.

Tuning Phrase-Based Segmented Translation for a Morphologically Complex Target Language

Stig-Arne Grönroos
Department of Signal Processing and Acoustics
Aalto University, Finland
stig-arne.gronroos@aalto.fi

Sami Virpioja
Department of Computer Science
Aalto University, Finland
sami.virpioja@aalto.fi

Mikko Kurimo
Department of Signal Processing and Acoustics
Aalto University, Finland
mikko.kurimo@aalto.fi

Abstract

This article describes the Aalto University entry to the English-to-Finnish shared translation task in WMT 2015. The system participates in the constrained condition, but in addition we impose some further constraints, using no language-specific resources beyond those provided in the task. We use a morphological segmenter, Morfessor FlatCat, but train and tune it in an unsupervised manner. The system could thus be used for another language pair with a morphologically complex target language, without needing modification or additional resources.

1 Introduction

In isolating languages, such as English, suitable smallest units of translation are easy to find using whitespace and punctuation characters as delimiters. This approach of using words as the smallest unit of translation is problematic for synthetic languages with rich inflection, derivation or compounding. Such languages have very large vocabularies, leading to sparse statistics and many out-of-vocabulary words.

A synthetic language uses fewer words than an isolating language to express the same sentence, by combining several grammatical markers into each word and using compound words. This difference in granularity is problematic in alignment, when a word in the isolating language properly aligns with only a part of a word in the synthetic language.

In order to balance the number of tokens between target and source, it is often possible to segment the morphologically richer side. Oversegmentation is detrimental, however, as longer windows of history need to be used, and useful phrases become more difficult to extract. It is therefore important to find a balance in the amount of segmentation. A linguistically accurate segmentation may be oversegmented for the task of translation, if some of the distinctions are either unmarked or marked in a similar way in the other language.

An increase in the number of tokens means that the distance spanned by dependencies becomes longer. Recurrent Neural Network (RNN) based language models have been shown to perform well for English (Mikolov et al., 2011). Their strength lies in being theoretically capable of modeling arbitrarily long dependencies.

Moreover, a huge vocabulary is particularly detrimental for neural language models due to their computationally heavy training and need to marginalize over the whole vocabulary during prediction. As morphological segmentation can reduce the vocabulary size considerably, using RNN language models seems even more suitable for this approach.

Our system is designed for translation in the direction from a morphologically less complex to a more complex language. The opposite direction – simplifying morphology – has received more attention, especially with English as the target language.

Of the target languages in this year's task, Finnish is the most difficult to translate into, shown by Koehn (2005) and reconfirmed by the evaluations of this shared task. Even though the use of supervised linguistic tools

Proceedings of the Tenth Workshop on Statistical Machine Translation, pages 105–111,
Lisboa, Portugal, 17-18 September 2015.

(such as taggers, parsers, or morphological analyzers) was allowed in the constrained condition, our method does not use them. It is therefore applicable to other morphologically complex target languages.

1.1 Related work

The idea of transforming morphology to improve statistical machine translation (SMT) is well established in the literature. An early example is Nießen and Ney (2004), who apply rule-based morphological analysis to enhance German→English translation.

In particular, many efforts have focused on increasing the symmetry between languages in order to improve alignment. Lee (2004) uses this idea for Arabic→English translation. In this translation direction, symmetry is increased through morphological simplification.

It has been shown that a linguistically correct segmentation does not coincide with the optimal segmentation for purposes of alignment, both using rule-based simplification of linguistic analysis (Habash and Sadat, 2006), and through the use of statistical methods (Chung and Gildea, 2009).

Using segmented translation with unsupervised statistical segmentation methods has yielded mixed results. Virpioja et al. (2007) used Morfessor Categories-MAP in translation between three Nordic languages, including Finnish, while Fishel and Kirik (2010) used Morfessor Categories-MAP in English↔Estonian translation. In these studies, segmentation has in many cases worsened BLEU compared to word-based translation. The main benefit of segmentation has been a decrease in the ratio of untranslated words.

Salameh et al. (2015) translate English→Arabic, and find that segmentation is most useful when the extracted phrases are morphologically productive, and that using a word-level language model reduces this productivity (albeit increasing the BLEU score).

The desegmentation process, and the effect of different strategies for marking the word-internal token boundaries, have mostly been examined in recombining split compound words. Stymne and Cancedda (2011) explore different marking strategies, including use of part-of-speech tags, in order to allow the translation system to produce compounds unseen in the training data.

2 System overview

An overview of the system is shown in Figure 1. The four main contributions of this work are indicated by numbered circles:

1. Use of unsupervised Morfessor FlatCat (Grönroos et al., 2014) for morphological segmentation,
2. Tuning the morphological segmentation directly to balance the number of translation tokens between source and target,
3. A new marking strategy for morph boundaries,
4. Rescoring n-best lists with RNNLM (Mikolov et al., 2010).

Our system extends an existing phrase-based SMT system to perform segmented translation, by adding pre-processing and post-processing steps, with no changes to the decoder. As translation system to be extended, we used the Moses release 3.0 (Koehn et al., 2007). We used GIZA++ alignment, and a 5-gram LM with modified-KN smoothing. Many Moses settings were left at their default values: phrase length 10, grow-diag-final-and alignment symmetrization, msd-bidirectional-fe reordering, and distortion limit 6.

The standard pre-processing steps not specified in Figure 1 consist of normalization of punctuation, tokenization, and statistical truecasing. All three of these were performed with the tools included in Moses.

In addition, the parallel data was cleaned and duplicate sentences were removed. Cleaning was performed after morphological segmentation, as the segmentation can increase the length in tokens of a sentence.

The post-processing steps are the reverse of the pre-processing steps: desegmentation, detruecasing, and detokenization. Rescoring of the n-best list was done before post-processing.

The feature weights were tuned using MERT (Och, 2003), with BLEU (Papineni et al., 2002) of the post-processed hypothesis

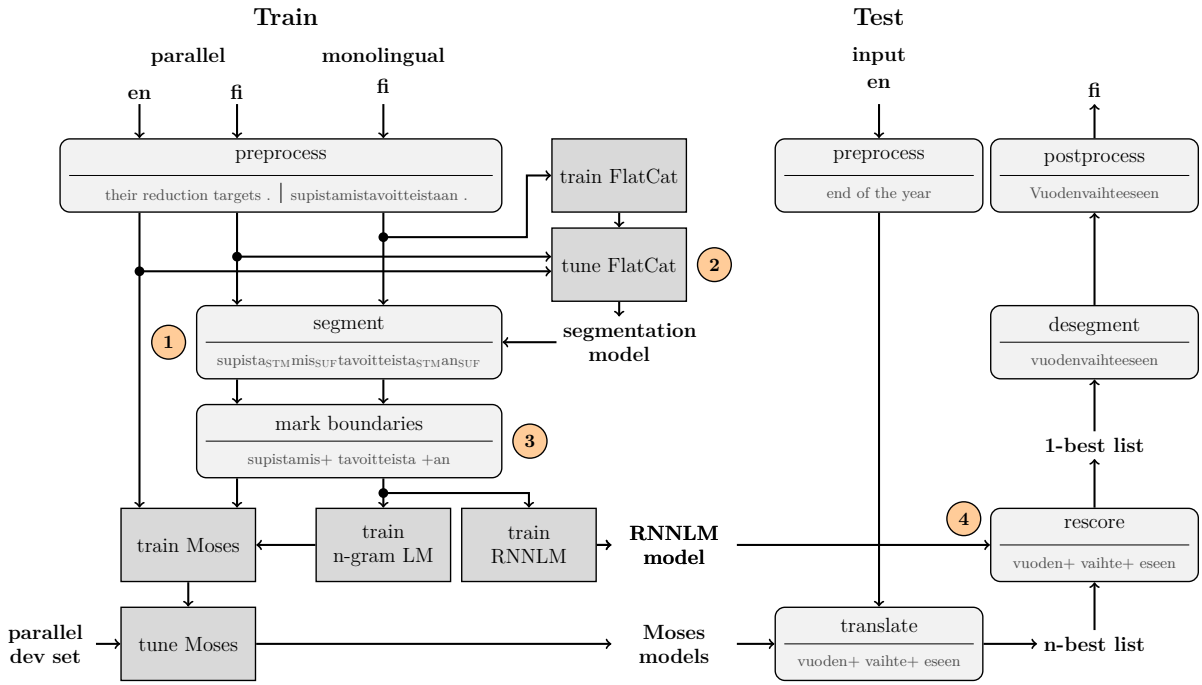


Figure 1: A pipeline overview of training and testing of the system. Main contributions are hilighted with numbers 1-4.

against a tuning set as the metric. 20 random restarts per MERT iteration were used, with iterations repeated until convergence.

A similar MERT procedure was also used for choosing the interpolation weights for rescoring, with 100 random restarts in a single iteration. A single-iteration approach was chosen, as there was no need to translate a new n-best list during the MERT for rescoring.

2.1 Morphological segmentation

For morphological segmentation, we use the latest Morfessor variant, FlatCat (Grönroos et al., 2014). Morfessor FlatCat is a probabilistic method for learning morphological segmentations, using a prior over morph lexicons inspired by the Minimum Description Length principle (Rissanen, 1989).

Morfessor FlatCat applies a Hidden Markov model for morphotactics. Compared to Morfessor Baseline, it provides morph category tags (stem, prefix, suffix) and has superior consistency especially in compound word splitting. In contrast to Categories-MAP (Creutz and Lagus, 2005), used for statistical machine translation e.g. by Clifton and Sarkar (2011), it supports semi-supervised learning and hyper-parameter tuning.

No annotated data was used in the training of Morfessor FlatCat, neither in training nor parameter tuning. Instead of aiming for a linguistic morphological segmentation, our goal was to balance the number of translation tokens between source and target languages.

In order to bring the number of tokens on the Finnish target side closer to the English source side, we segmented the Finnish text with an unsupervised Morfessor FlatCat model, tuned specifically to achieve this balance. The corpus weight hyper-parameter α was chosen by minimizing the sentence-level difference in token counts between the English and the segmented Finnish sides of the parallel corpus

$$\alpha = \arg\min_{\alpha} \sum_{(e,f)\in(E,F)} \Big|\#(e) - \#\big(M(f;\alpha)\big)\Big|, \tag{1}$$

where # gives the number of tokens in the sentence, and $M(f;\alpha)$ is the segmentation with a particular α.

Numbers and URLs occurring in the parallel corpus were passed through Morfessor unseg-

mented, but translated by Moses without any special handling.

2.2 Morph boundary marking strategy

In the desegmentation step, consecutive tokens are concatenated either with or without an intermediary space. Morph boundaries must be distinguished from word boundaries, so that the desegmentation step can reconstruct the words correctly. There are various ways to mark the boundaries, some of them shown in Table 1.

A common way is to attach a symbol to all morphs on the right (or left) side of the morph boundary. We call this strategy *right-only*.

Alternatively *both-sides* of the boundary can be marked. In this strategy, a decision must be made whether to be aggressive or conservative in joining morphs, if the translation system outputs an incorrect sequence where the markers do not match up on both sides. For these experiments we chose the conservative approach, removing the unmatched marker from a half-marked boundary, and treating it as a word boundary.

A downside of the *right-only* and *both-sides* strategies is that a stem is marked differently depending on whether it has a prefix attached or not, even if the surface form of the stem does not change.

The morph categories produced by FlatCat can be used for marking boundaries according to the structure of the word. We can mark affixes from the side that points towards the stem, leaving stems unmarked regardless of the presence of affixes. However, this would leave the boundaries between compound parts indistinguishable from word boundaries, making some additional marking necessary.

Marking affixes by category and compound boundaries with a special linking token is called the *compound-symbol* strategy. Instead marking the last morpheme in the compound modifiers (non-final compound parts), results in the *compound-left* strategy.

After initial unimpressive results with the compound marking strategies, we concluded that segmenting the compound modifiers does not lead to productive translation phrases, in contrast to boundaries between compound parts and boundaries separating inflective affixes. In response, we formulated the *advanced* marking strategy, which goes beyond boundary marking to modify the segmentation, by rejoining the morphs in the modifier parts of compounds.

Strategy	Example
Surface form	supistamistavoitteistaan
Segmentation	$\text{supista}_{\text{STM}}\text{mis}_{\text{SUF}}\text{tavoitteista}_{\text{STM}}\text{an}_{\text{SUF}}$
Translation	of their reduction targets
right-only	supista +mis +tavoitteista +an
both-sides	supista+ +mis+ +tavoitteista+ +an
compound-sym	supista +mis +@+ tavoitteista +an
compound-left	supista +mis@ tavoitteista +an
advanced	supistamis+ tavoitteista +an

Table 1: Morph boundary marking strategies.

The sequence of morph categories is used for grouping the morphs into compound parts. A word consists of one or more compound parts. Each compound part consists of exactly one stem, and any number of preceding prefixes and following suffixes.

$$\begin{aligned} \textsc{CompoundPart} &= \textsc{Pre}^* \ \textsc{Stm} \ \textsc{Suf}^* \\ \textsc{Word} &= \textsc{CompoundPart}^+ \end{aligned} \quad (2)$$

For all compound parts except the last one, the affixes are rejoined to their stem. Morphs of length 5 or above were treated as stems, regardless of the category assigned to them by FlatCat.

Prefixes and compound modifiers are marked with a trailing '+', suffixes are marked with a leading '+', and the stems of the word-final compound parts are left unmarked.

2.3 Rescoring n-best lists

Segmentation of the word forms increases the distances spanned by dependencies that should be modeled by the language model. To compensate this, we apply a strong recurrent neural network language model (RNNLM) (Mikolov et al., 2010). The additional language model is used in a separate rescoring step, to speed up translation, and for ease of implementation.

The RNNLM model was trained on morphologically segmented data. Morphs occurring only once were removed from the vocabulary, and replaced with <UNK>. The parameters were set to 300 nodes in the hidden layer, 500 vocabulary classes, 2M direct connections of

	Monolingual data	Parallel data			
Purpose	news2014 v2	europarl v8	wikititles	newsdev2015	test2006
Training Morfessor	fi	fi	fi		
Training LMs	fi	fi	fi		
Training Moses		en – fi	en – fi		
Tuning Morfessor		en – fi			
Tuning RNNLM				fi	
Tuning Moses				en – fi	
Development testing					en – fi
Sentences	1378582	1926114	153728	1500	2000

Table 2: The data sets used for different purposes. "en–fi" signifies that parallel data was used, "fi" signifies monolingual data, or using only the Finnish side of parallel data.

order 4, backpropagation through 5 time steps, with blocksize 25.

At translation time, 1000-best lists of morph segmented hypotheses produced by Moses were scored using the RNNLM.

The Moses features were extended by including the RNNLM score as an additional feature. A new linear combination of the features was optimized with MERT, and used for the final hypothesis ranking. For the BLEU measurement in MERT the segmented hypothesis was post-processed (including desegmentation) and compared to an un-preprocessed reference.

3 Data

The data sets used in training and tuning are shown in Table 2. Both *europarl v8* and *wikititles* were used as parallel training data, but only europarl was used for tuning the hyperparameter α, as the titles do not follow a typical sentence structure.

The Finnish side of the parallel sets was used to extend the monolingual training data. The monolingual data were concatenated for LM training, instead of interpolating different n-gram models.

After cleaning, the combined parallel training data contained 2,004,450 sentences. The parallel set used for testing during development is *test2006*, a europarl subset of 2000 sentences sampled from three last months of 2000.[1]

[1] `http://matrix.statmt.org/test_sets/list`

Configuration	dev-test test2006 BLEU	test newstest2015 BLEU
advanced, $\alpha = 0.7$	**.147**	.112
+rescoring	**.147**	**.116**
advanced, $\alpha = 0.4$	.145	.112
both-sides	.141	.114
compound-left	.140	.113
compound-sym	.139	.111
right-only	.139	.111
(word)	.146	.100

Table 3: Results of evaluation.

4 Results

Table 3 shows cased BLEU scores on the in-domain development set and out-of-domain test set, for various configurations. The entry marked *word* is a baseline system without segmentation.

When evaluating on the in-domain development set, most configurations that use segmentation achieve worse BLEU compared to the word baseline. Only the best configurations, using the *advanced* strategy, are able to achieve slightly higher BLEU.

Switching domains to the test corpus leads to a larger difference, in favor of the segmenting methods. The choice of morph boundary marking strategy and the sentence-based tuning of the segmentation had a moderate effect on BLEU. The addition of rescoring did not improve BLEU on the in-domain dev-test corpus, but resulted in a slight improvement on

the out-of-domain test corpus.

The proportion of word tokens that were segmented into at least two parts was 19.8%. The joining of compound modifiers did not have a large effect on the total number of tokens, causing a reduction from 49,524,520 to 49,475,291 (0.1%).

Using the sentence-level balancing, the optimal value for the corpus weight hyperparameter α was 0.7. The change in the number of tokens caused by the joining of compound modifiers did not affect the optimum. Balancing the token count of the whole corpus yielded a much lower α of 0.4, leading to oversegmentation and lower BLEU.

The weight of the RNNLM in the final linear combination was 0.092, compared to 0.119 of the n-gram LM. This indicates that it is able to complement the n-gram model, but does not dominate it.

In the human evaluation of WMT15, the system with advanced morph boundary marking strategy and RNNLM rescoring was ranked in tied second place of five methods participating in the constrained condition.

5 Conclusions

To improve English-to-Finnish translation in a phrase-based machine translation system, we tuned an unsupervised morphological segmentation preprocessor to balance the token count between source and target languages. Appropriate choice of morph boundary marking strategy and amount of segmentation brought the BLEU score slightly above a word-based baseline, in contrast to some previous work with unsupervised segmentation (Virpioja et al., 2007; Fishel and Kirik, 2010).

To compensate for the need of longer contexts, we added a recurrent neural network language model as a rescoring step. It did not help for the in-domain development corpus, but improved results on the out-of-domain test corpus.

Possible directions for future work include Minimum Bayes Risk combination of translation hypotheses from systems trained with different segmentations and marking strategies (De Gispert et al., 2009), using morphology generation instead of segmented translation (Clifton and Sarkar, 2011), and improving the alignment directly in addition to balancing of token counts (Snyder and Barzilay, 2008).

Acknowledgments

This research has been supported by the Academy of Finland under the Finnish Centre of Excellence Program 2012–2017 (grant n°251170), and LASTU Programme (grants n°256887 and 259934). Computer resources within the Aalto University School of Science "Science-IT" project were used.

References

[Chung and Gildea2009] Tagyoung Chung and Daniel Gildea. 2009. Unsupervised tokenization for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 718–726. Association for Computational Linguistics.

[Clifton and Sarkar2011] Ann Clifton and Anoop Sarkar. 2011. Combining morpheme-based machine translation with post-processing morpheme prediction. In *Proceedings of ACL-HLT*, pages 32–42. Association for Computational Linguistics.

[Creutz and Lagus2005] Mathias Creutz and Krista Lagus. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In Timo Honkela, Ville Könönen, Matti Pöllä, and Olli Simula, editors, *Proceedings of AKRR'05*, pages 106–113, Espoo, Finland, June. Helsinki University of Technology, Laboratory of Computer and Information Science.

[De Gispert et al.2009] Adrià De Gispert, Sami Virpioja, Mikko Kurimo, and William Byrne. 2009. Minimum Bayes risk combination of translation hypotheses from alternative morphological decompositions. In *Proceedings of HLT-NAACL 2009: Short Papers*, pages 73–76. Association for Computational Linguistics.

[Fishel and Kirik2010] Mark Fishel and Harri Kirik. 2010. Linguistically motivated unsupervised segmentation for machine translation. In *LREC*.

[Grönroos et al.2014] Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. 2014. Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, pages 1177–1185. Association for Computational Linguistics.

[Habash and Sadat2006] Nizar Habash and Fatiha Sadat. 2006. Arabic preprocessing schemes for statistical machine translation. In *Proceedings of HLT-NAACL*. Association for Computational Linguistics.

[Koehn et al.2007] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

[Koehn2005] Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

[Lee2004] Young-Suk Lee. 2004. Morphological analysis for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 57–60. Association for Computational Linguistics.

[Mikolov et al.2010] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

[Mikolov et al.2011] Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan Honza Cernocky. 2011. Empirical evaluation and combination of advanced language modeling techniques. In *Interspeech*. ISCA, August.

[Nießen and Ney2004] Sonja Nießen and Hermann Ney. 2004. Statistical machine translation with scarce resources using morpho-syntactic information. *Computational linguistics*, 30(2):181–204.

[Och2003] Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

[Papineni et al.2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

[Rissanen1989] Jorma Rissanen. 1989. *Stochastic Complexity in Statistical Inquiry*, volume 15. World Scientific Series in Computer Science, Singapore.

[Salameh et al.2015] Mohammad Salameh, Colin Cherry, and Grzegorz Kondrak. 2015. What matters most in morphologically segmented SMT models? *Syntax, Semantics and Structure in Statistical Translation*, page 65.

[Snyder and Barzilay2008] Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *ACL*, pages 737–745.

[Stymne and Cancedda2011] Sara Stymne and Nicola Cancedda. 2011. Productive generation of compound words in statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 250–260. Association for Computational Linguistics.

[Virpioja et al.2007] Sami Virpioja, Jaakko J Väyrynen, Mathias Creutz, and Markus Sadeniemi. 2007. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. *Machine Translation Summit XI*, 2007:491–498.

Publication VIII

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. Cognate-aware morphological segmentation for multilingual neural translation. In *Proceedings of the Third Conference on Machine Translation*, Brussels, Belgium, pages 386–393, Oct 2018.

Cognate-aware morphological segmentation for multilingual neural translation

Stig-Arne Grönroos
stig-arne.gronroos@aalto.fi
Aalto University, Finland

Sami Virpioja
sami.virpioja@aalto.fi
Aalto University, Finland
Utopia Analytics, Finland

Mikko Kurimo
mikko.kurimo@aalto.fi
Aalto University, Finland

Abstract

This article describes the Aalto University entry to the WMT18 News Translation Shared Task. We participate in the multilingual subtrack with a system trained under the constrained condition to translate from English to both Finnish and Estonian. The system is based on the Transformer model. We focus on improving the consistency of morphological segmentation for words that are similar orthographically, semantically, and distributionally; such words include etymological cognates, loan words, and proper names. For this, we introduce Cognate Morfessor, a multilingual variant of the Morfessor method. We show that our approach improves the translation quality particularly for Estonian, which has less resources for training the translation model.

1 Introduction

Cognates are words in different languages, which due to a shared etymological origin are represented as identical or nearly identical strings, and also refer to the same or similar concepts. Ideally the cognate pair is similar orthographically, semantically, and distributionally. Care must be taken with "false friends", i.e. words with similar string representation but different semantics. Following usage in Natural Language Processing, e.g. (Kondrak, 2001), we use this broader definition of the term cognate, without placing the same weight on etymological origin as in historical linguistics. Therefore we accept loan words as cognates.

In any language pair written in the same alphabet, cognates can be found among names of persons, locations and other proper names. Cognates are more frequent in related languages, such as Finnish and Estonian. These additional cognates are words of any part-of-speech, which happen to have a shared origin.

In this work we set out to improve morphological segmentation for multilingual translation systems with one source language and two related target languages. One of the target languages is assumed to be a low-resource language. The motivation for using such a system is to exploit the large resources of a related language in order to improve the quality of translation into the low-resource language.

Consistency of the segmentations is important when using subword units in machine translation. We identify three types of consistency in the multilingual translation setting (see examples in Table 1):

(i) The benefit of consistency is most evident when the translated word is an identical cognate between the source and a target language. If the source and target segmentations are consistent, such words can be translated by sequentially copying subwords from source to target.

(ii) Language-internal consistency means that when a subword boundary is added, its location corresponds to a true morpheme boundary, and that if some morpheme boundaries are left unsegmented, the choices are consistent between words. This improves the productivity of the subwords and reduces the risk of introducing short, word-internal errors at the subword boundaries. In the example **saami + miseksi*, choosing the wrong second morph causes the letters *mi* to be accidentally repeated.

(iii) When training a multilingual model, a third form of consistency arises between the different target languages. An optimal segmentation would maximize the use of morphemes with cross-lingually similar string rep-

Proceedings of the Third Conference on Machine Translation (WMT), Volume 2: Shared Task Papers, pages 386–393

https://doi.org/10.18653/v1/W18-64037

type	consistent	en	fi	et
(i)	yes	On + y + sz + kie + wicz	On + y + sz + kie + wicz	On + y + sz + kie + wicz
(ii)	yes	gett + ing work + ing	saa + mise + ksi toimi + mise + ksi	saa + mise + ks toimi + mise + ks
(iii)	yes	work time	työ + aja + sta	töö + aja + st
(i)	no	On + y + sz + kie + wicz	Onys + zk + ie + wi + cz	O + nysz + ki + ewicz
(ii)	no	get + ting work + ing	saami + seksi toimi + mise + ksi	saami + seks toimi + miseks
(iii)	no	work time	työ + aja + sta	tööajast

Table 1: Example consistent and inconsistent segmentations.

resentations and meanings, whether they occur in cognate words or elsewhere. We hypothesize that segmentation consistency between target languages enables learning of better generalizing subword representations. This consistency allows contexts seen in the high-resource corpus to fill in for those missing from the low-resource corpus. This should lead to improved translation results, especially for the lower resourced target language.

Naïve joint training of a segmentation model, e.g. by training Byte Pair Encoding (BPE) (Sennrich et al., 2015) on the concatenation of the training corpora in different languages, can only address consistency when the cognates are identical (type *i*), or with some luck if the differences occur in the ends of the words. If a single letter changes in the middle of a cognate, consistent subwords that span over the location of the change are found only by chance. In order to encourage stronger consistency, we propose a segmentation model that uses automatically extracted cognates and fuzzy matching between cognate morphs.

In this work we also contribute two new features to the OpenNMT translation system: Ensemble decoding, and fine-tuning a pre-trained model using a compatible data set.[1]

1.1 Related work

Improving segmentation through multilingual learning has been studied before. Snyder and Barzilay (2008) propose an unsupervised, Bayesian method, which only uses parallel phrases as training data. Wicentowski (2004) present a supervised method, which requires lemmatization. The method of Naradowsky and Toutanova (2011) is also unsupervised, utilizing a hidden semi-Markov model, but it requires rich features on the input data.

The subtask of cognate extraction has seen much research effort (Mitkov et al., 2007; Bloodgood and Strauss, 2017; Ciobanu and Dinu, 2014). Most methods are supervised, and/or require rich features.

There is also work on cognate identification from historical linguistics perspective (Rama, 2016; Kondrak, 2009), where the aim is to classify which cognate candidates truly share an etymological origin.

We propose a language-agnostic, unsupervised method, which doesn't require annotations, lemmatizers, analyzers or parsers. Our method can exploit both monolingual and parallel data, and can use cognates of any part-of-speech.

2 Cognate Morfessor

We introduce a new variant of Morfessor for cross-lingual segmentation.[2] It is trained using a bilingual corpus, so that both target languages are trained simultaneously.

We allow each language to have its own subword lexicon. In essence, as a Morfessor model consists of a lexicon and the corpus encoded with that lexicon, we now have two separate complete Morfessor sub-models. The two models are linked through the training algorithm. We want the segmentation of non-cognates to tend towards the normal Morfessor Baseline segmentation, but place some additional constraints on how the cognates are segmented.

In our first experiments, we only restricted the number of subwords on both sides of the cognate pair to be equal. This criterion was

[1] Our changes are awaiting inclusion in OpenNMT. In the mean time, they are available from https://github.com/Waino/OpenNMT-py/tree/ensemble

[2] Available from https://github.com/Waino/morfessor-cognates

too loose, and we saw many of the longer cognates segmented with both 1-to-N and N-to-1 morpheme correspondences. For example

ty	+	ö	+	aja	+	sta
töö	+	aja	+	s	+	t

To further encourage consistency, we included a third component to the model, which encodes the letter edits transforming the subwords of one cognate into the other.

Cognate Morfessor is inspired by Allomorfessor (Kohonen et al., 2009; Virpioja et al., 2010), which is a variant of Morfessor that includes modeling of allomorphic variation. Simultaneously to learning the segmentations, Allomorfessor learns a lexicon of transformations to convert a morph into one of its allomorphs. Allomorfessor is trained on monolingual data.

We implement the new version as an extension of Morfessor Baseline 2.0 (Virpioja et al., 2013).

2.1 Model

The Morfessor Baseline cost function (Creutz and Lagus, 2002)

$$\mathrm{L}(\boldsymbol{\theta}, \boldsymbol{D}) = -\log p(\boldsymbol{\theta}) - \log p(\boldsymbol{D} \mid \boldsymbol{\theta}) \qquad (1)$$

is extended to

$$\begin{aligned}\mathrm{L}(\boldsymbol{\theta}, \boldsymbol{D}) = &-\log p(\boldsymbol{\theta}_1) - \log p(\boldsymbol{\theta}_2) - \log p(\boldsymbol{\theta}_E) \\ &- \log p(\boldsymbol{D}_1 \mid \boldsymbol{\theta}_1) - \log p(\boldsymbol{D}_2 \mid \boldsymbol{\theta}_2) \\ &- \log p(\boldsymbol{D}_E \mid \boldsymbol{\theta}_E) \qquad (2)\end{aligned}$$

dividing both lexicon and corpus coding costs into three parts: one for each language ($\boldsymbol{\theta}_1, \boldsymbol{D}_1$ and $\boldsymbol{\theta}_2, \boldsymbol{D}_2$) and one for the edits transforming the cognates from one language to the other ($\boldsymbol{\theta}_E, \boldsymbol{D}_E$).

The coding is redundant, as one language and the edits would be enough to reconstruct the second language. In the interest of symmetry between target languages, we ignore this redundancy.

The intuition is that the changes in spelling between the cognates in a particular language pair is regular. Coding the differences in a way that reduces the cost of making a similar change in another word guides the model towards learning these patterns from the data.

The coding of the edits is based on the Levenshtein (1966) algorithm. Let (w^a, w^b) be a cognate pair and its current segmentation $\big((m_1^a, \ldots, m_n^a), (m_1^b, \ldots m_n^b)\big)$. The morphs are paired up sequentially. Note that the restrictions on the search algorithm guarantee that both segmentations contain the same number of morphs, n. For a morph pair (m_i^a, m_i^b), the Levenshtein-minimal set of edits is calculated. Edits that are immediately adjacent to each other are merged. In order to improve the modeling of sound length change, we extend the edit in both languages to include the neighboring unchanged character, if one half of the edit is the empty string ϵ, and the other contains another instance of character representing the sound being lengthened or shortened. This extension encodes a sound lengthening as e.g. 'a→aa' instead of 'ϵ →a'. As the edits are cheaper to reuse once added to the edit lexicon, avoiding edits with ϵ on either side is beneficial to reduce spurious use. Finally, position information is discarded from the edits, leaving only the substrings, separated by a boundary symbol.

As an example, the edits found between *yhteenkuuluvuuspolitiikkaa* and *ühtekuuluvuspoliitika* are 'y→ü', 'een→e', 'uu→u', 'ti→it', and 'kka→k'.

The semi-supervised weighting scheme of Kohonen et al. (2010) can be applied to Cognate Morfessor. A new weighting parameter *edit_cost_weight* is added, and multiplicatively applied to both the lexicon and corpus costs of the edits.

The training algorithm is an iterative greedy local search very similar to the Morfessor Baseline algorithm. The algorithm finds an approximately minimizing solution to Eq 2. The recursive splitting algorithm from Morfessor Baseline is slightly modified. If a non-cognate is being reanalyzed, the normal algorithm is followed. Cognates are reanalyzed together. Recursive splitting is applied, with the restriction that if a morph in one language is split, then the corresponding cognate morph in the other language must be split as well. The Cartesian product of all combinations of valid split points for both languages is tried, and the pair of splits minimizing the cost function is selected, unless not splitting results in even lower cost.

3 Extracting cognates from parallel data

Finnish–Estonian cognates were automatically extracted from the shared task training data. As we needed a Finnish–Estonian parallel data set, we generated one by triangulation from the English–Finnish and English–Estonian parallel data. This resulted in a set of 679 252 sentence pairs (ca 12 million tokens per language).

FastAlign (Dyer et al., 2013) was used for word alignment in both directions, after which the alignments were symmetrized using the *grow-diag-final-and* heuristic. All aligned word pairs were extracted based on the symmetrized alignment. Words containing punctuation, and pairs aligned to each other fewer than 2 times were removed. The list of word pairs was filtered based on Levenshtein distance. If either of the words consisted of 4 or fewer characters, an exact match was required. Otherwise, a Levenshtein distance up to a third of the mean of the lengths, rounding up, was allowed. This procedure resulted in a list of 40 472 cognate pairs. The list contains words participating in multiple cognate pairs. Cognate Morfessor is only able to link a word to a single cognate. We filtered the list, keeping only the pairing to the most frequent cognate, which reduces the list to 22 226 pairs.

The word alignment provides a check for semantic similarity in the form of translational equivalence. Even though the word alignment may produce some errors, accidentally segmenting false friends consistently should not be problematic.

4 Data

After filtering, we have 9 million multilingual sentence pairs in total. 6.3M of this is English–Finnish, of which 2.2M is parallel data, and 4.1M is synthetic backtranslated data. Of the 2.8M total English–Estonian, 1M is parallel and 1.8M backtranslated. The sentences backtranslated from Finnish were from the news.2016.fi corpus, translated with a PB-SMT model, trained with WMT16 constrained settings. The backtranslation from Estonian was freshly made with a BPE-based system similar to our baseline system, trained on the WMT18 data. The sentences were selected from the news.20{14-17}.et corpora, using a language model filtering technique.

4.1 Preprocessing

The preprocessing pipeline consisted of filtering by length[3] and ratio of lengths[4], fixing encoding problems, normalizing punctuation, removing of rare characters[5], deduplication, tokenizing, truecasing, rule-based filtering of noise, normalization of contractions, and filtering of noise using a language model.

The language model based noise filtering was performed by training a character-based deep LSTM language model on the in-domain monolingual data, using it to score each target sentence in the parallel data, and removal of sentences with perplexity per character above a manually picked threshold. A lenient threshold[6] was selected in order to filter noise, rather than for aiming for domain adaptation. The same process was applied to filter the Estonian news data for backtranslation.

Our cognate segmentation resulted in a target vocabulary of 42 386 subwords for Estonian and 46 930 subwords for Finnish, resulting in 64 396 subwords when combined.

For segmentation of the English source, a separate Morfessor Baseline model was trained. To ensure consistency between source and target segmentations, we used the segmentation of the Cognate Morfessor model for any English words that were also present in the target side corpora. The source vocabulary consisted of 61 644 subwords.

As a baseline segmentation, we train a shared 100k subword vocabulary using BPE. To produce a balanced multilingual segmentation, the following procedure was used: First, word counts were calculated individually for English and each of the target languages Finnish and Estonian. The counts were normalized to equalize the sum of the counts for each language. This avoided imbalance in the amount of data skewing the segmentation in favor of some language. BPE was trained on the balanced counts. Segmentation boundaries around hyphens were forced, overriding the BPE.

[3] 1–100 tokens, 3–600 chars, $\leq$ 50 chars/token.
[4] Requiring ratio 0.5–2.0, if either side $>$ 10 chars.
[5] $<$ 10 occurrences
[6] 96% of the data was retained.

ϵ	→	n	27919	g	→	k	3000	il	→	ϵ	2077
ϵ	→	a	17082	ü	→	y	2979	m	→	mm	2016
ϵ	→	i	15725	oo	→	o	2790	s	→	n	2005
d	→	t	12599	t	→	a	2674	ee	→	e	1950
l	→	ll	5236	ϵ	→	k	2583	i	→	ϵ	1889
ϵ	→	ä	4437	aa	→	a	2536	ϵ	→	e	1803
s	→	ssa	3907	õ	→	o	2493	u	→	o	1724
t	→	tt	3863	a	→	ä	2479	ϵ	→	d	1496
o	→	u	3768	s	→	ϵ	2173	il	→	t	1486
e	→	i	3182	t	→	ϵ	2158	d	→	ϵ	1433

Table 2: 30 most frequent edits learned by the model. The direction is Estonian→Finnish. The numbers indicate how many times the edit was applied in the morph lexicon. ϵ indicates the empty string.

EN-ET	chrF-1.0 dev	BLEU% dev
BPE	56.52	17.93
monolingual	53.44	15.82
Cognate Morfessor	57.05	18.40
+finetuned	57.23	18.45
+ensemble-of-5	**57.75**	**19.09**
+ensemble-of-3	57.64	18.96
+linked embeddings	56.20	17.48
−LM filtering	52.94	14.65
6+6 layers	57.35	18.84

Table 3: Development set results for English–Estonian. character-F and BLEU scores in percentages. +/− stands for adding/removing a component. Multiple modifications are indicated by increasing the indentation.

Multilingual translation with target-language tag was done following (Johnson et al., 2016). A pseudo-word, e.g. <TO_ET> to mark Estonian as the target language, was prefixed to each paired English source sentence.

5 NMT system

We use the OpenNMT-py (Klein et al., 2017) implementation of the Transformer.

5.1 Transformer

The Transformer architecture (Vaswani et al., 2017) relies fully on attention mechanisms, without need for recurrence or convolution. A Transformer is a deep stack of layers, consisting of two types of sub-layer: multi-head (MH) attention (Att) sub-layers and feed-forward (FF) sub-layers:

$$\begin{aligned} \mathrm{Att}(Q,K,V) &= \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V \\ a_i &= \mathrm{Att}(QW_i^Q, KW_i^K, VW_i^V) \\ \mathrm{MH}(Q,K,V) &= [a_1;\ldots;a_h]W^O \\ \mathrm{FF}(x) &= \max(0, xW_1 + b_1)W_2 + b_2 \end{aligned} \quad (3)$$

where Q is the input query, K is the key, and V the attended values. Each sub-layer is individually wrapped in a residual connection and layer normalization.

When used in translation, Transformer layers are stacked into an encoder-decoder structure. In the encoder, the layer consists of a self-attention sub-layer followed by a FF sub-layer. In self-attention, the output of the previous layer is used as queries, keys and values $Q = K = V$. In the decoder, a third context attention sub-layer is inserted between the self-attention and the FF. In context attention, Q is again the output of the previous layer, but $K = V$ is the output of the encoder stack. The decoder self-attention is also masked to prevent access to future information. Sinusoidal position encoding makes word order information available.

5.2 Training

Based on some preliminary results, we decided to reduce the number of layers to 4 in both encoder and decoder; later we found that the decision was based on too short training time. Other parameters were chosen following the OpenNMT FAQ (Rush, 2018): 512-dimensional word embeddings and hidden states, dropout 0.1, batch size 4096 tokens, label smoothing 0.1, Adam with initial learning rate 2 and β_2 0.998.

Fine-tuning for each target language was performed by continuing training of a multilingual model. Only the appropriate monolingual subset of the training data was used in this phase. The data was still prefixed for target language as during multilingual training. No vocabulary pruning was performed.

In our ensemble decoding procedure, the predictions of 3–8 models are combined by averaging after the softmax layer. Best results are achieved when the models have been independently trained. However, we also try combinations where a second copy of a model is further trained with a different configuration (monolingual finetuning).

	chrF-1.0				BLEU%			
EN-FI	nt2015	nt2016	nt2017	nt2017AB	nt2015	nt2016	nt2017	nt2017AB
BPE	58.59	59.76	62.00	63.06	21.09	21.04	23.49	26.55
monolingual	57.94	59.11	61.33	62.41	20.87	20.70	23.11	26.12
Cognate Morfessor	58.18	59.81	62.15	63.24	20.73	21.18	23.37	26.26
+finetuned	58.48	59.89	62.17	63.28	21.08	21.41	23.45	26.52
+ensemble-of-8	**59.07**	**60.69**	**62.94**	**64.07**	**21.50**	**22.34**	**24.59**	**27.55**
−LM filtering	58.19	59.39	61.78	62.82	20.62	20.77	23.38	26.36
+linked embeddings	57.79	59.45	61.52	62.58	19.95	20.84	22.70	25.69
6+6 layers	58.68	60.26	62.37	63.52	21.05	21.81	23.93	27.08

Table 4: Results for English–Finnish. character-F and BLEU scores in percentages. +/− stands for adding/removing a component. Newstest is abbreviated nt. Both references are used in nt2017AB.

We experimented with partially linking the embeddings of cognate morphs. In this experiment, we used morph embeddings concatenated from two parts: a part consisting of normal embedding of the morph, and a part that was shared between both halves of the cognate morph pair. Non-cognate morphs used an unlinked embedding also for the second part. After concatenation, the linked embeddings have the same size as the baseline embeddings.

We evaluate the systems with cased BLEU using the mteval-v13a.pl script, and characterF (Popovic, 2015) with β set to 1.0. The latter was used for tuning.

6 Results

Based on preliminary experiments, the Morfessor corpus cost weight α was set to 0.01, and the edit cost weight was set to 10. The most frequent edits are shown in Table 2.

Table 3 shows the development set results for Estonian. Table 4 shows results for previous year's test sets for Finnish.

The tables show our main system and the two baselines: a multilingual model using joint BPE segmentation, and a monolingual model using Morfessor Baseline.

Cognate Morfessor outperforms the comparable BPE system according to both measures for Estonian, and according to chrF-1.0 for Finnish. For Finnish, results measured with BLEU vary between test sets. The cross-lingual segmentation is particularly beneficial for Estonian.

In the monolingual experiment, the cross-lingual segmentations are replaced with monolingual Morfessor Baseline segmentation, and only the data sets of one language pair at a time is used. These results show that even the higher resourced language, Finnish, benefits from multilingual training.

The indented rows show variant configurations of our main system. Monolingual finetuning consistently improves results for both languages. For Estonian, we have two ensemble configurations: one combining 3 monolingually finetuned independent runs, and one combining 5 monolingually finetuned savepoints from 4 independent runs. Selection of savepoints for the ensemble was based on development set chrF-1. In the ensemble-of-5, one training run contributed two models: starting finetuning from epochs 14 and 21 of the multi-lingual training. The submitted system is the ensemble-of-3, as the ensemble-of-5 finished training after the deadline. For Finnish, we use an ensemble of 4 finetuned and 4 non-finetuned savepoints from 4 independent runs.

To see if further cross-lingual learning could be achieved, we performed an unsuccessful experiment with linked embeddings. It appears that explicit linking does not improve the morph representations over what the translation model is already capable of learning.

After the deadline, we trained a single model with 6 layers in both the encoder and decoder. This configuration consistently improves results compared to the submitted system.

All the variant configurations (ensemble, finetuning, LM filtering, linked embeddings, number of layers) used with Cognate Morfessor are compatible with each other. We did not not explore the combinations in this work, except for combining finetuning with ensembleing: all of the models in the Estonian ensembles, and 4 of the models in the Finnish

ensemble are finetuned. All the variant configurations except for linked embeddings could also be used with BPE.

7 Conclusions and future work

The translation system trained using the Cognate Morfessor segmentation outperforms the baselines for both languages. The benefit is larger for Estonian, the language with less data in this experiment.

One downside is that, due to the model structure, Cognate Morfessor is currently not applicable to more than two target languages.

Cognate Morfessor itself learns to model the frequent edits between cognate pairs. However, in the preprocessing cognate extraction step of this work, we used unweighted Levenshtein distance, which does not distinguish edits by frequency. In future work, weighted or graphonological Levenshtein distance could be applied (Babych, 2016).

Acknowledgments

This research has been supported by the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 780069. Computer resources within the Aalto University School of Science "Science-IT" project were used. We wish to thank Peter Smit for groundlaying work that led to Cognate Morfessor.

References

Bogdan Babych. 2016. Graphonological Levenshtein edit distance: Application for automated cognate identification. *Baltic Journal of Modern Computing*, 4(2):115–128.

Michael Bloodgood and Benjamin Strauss. 2017. Using global constraints and reranking to improve cognates detection. In *Proc. ACL*, volume 1, pages 1983–1992.

Alina Maria Ciobanu and Liviu P Dinu. 2014. Automatic detection of cognates using orthographic alignment. In *Proc. ACL*, volume 2, pages 99–105.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proc. SIGPHON*, pages 21–30, Philadelphia, PA, USA. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proc. NAACL*.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Oskar Kohonen, Sami Virpioja, and Mikaela Klami. 2009. Allomorfessor: Towards unsupervised morpheme analysis. In *Evaluating Systems for Multilingual and Multimodal Information Access*, volume 5706 of *Lecture Notes in Computer Science*, pages 975–982. Springer Berlin / Heidelberg.

Oskar Kohonen, Sami Virpioja, and Krista Lagus. 2010. Semi-supervised learning of concatenative morphology. In *Proc. SIGMORPHON*, pages 78–86, Uppsala, Sweden. Association for Computational Linguistics.

Grzegorz Kondrak. 2001. Identifying cognates by phonetic and semantic similarity. In *Proc. NAACL*, pages 1–8. Association for Computational Linguistics.

Grzegorz Kondrak. 2009. Identification of cognates and recurrent sound correspondences in word lists. *TAL*, 50(2):201–235.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Ruslan Mitkov, Viktor Pekar, Dimitar Blagoev, and Andrea Mulloni. 2007. Methods for extracting and classifying pairs of cognates and false friends. *Machine translation*, 21(1):29.

Jason Naradowsky and Kristina Toutanova. 2011. Unsupervised bilingual morpheme segmentation and alignment with context-rich hidden semi-Markov models. In *Proc. ACL: HLT*, pages 895–904. Association for Computational Linguistics.

Maja Popovic. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *WMT15*, pages 392–395.

Taraka Rama. 2016. Siamese convolutional networks for cognate identification. In *Proc. COLING*, pages 1018–1027.

Alexander M. Rush. 2018. OpenNMT FAQ – How do i use the Transformer model? `http://opennmt.net/OpenNMT-py/FAQ.html#how-do-i-use-the-transformer-model`. Accessed: 27.7.2018.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. In *Proc. ACL16*.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proc. ACL: HLT*, pages 737–745.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. NIPS*, pages 6000–6010.

Sami Virpioja, Oskar Kohonen, and Krista Lagus. 2010. Unsupervised morpheme analysis with Allomorfessor. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments*, volume 6241 of *Lecture Notes in Computer Science*, pages 609–616. Springer Berlin / Heidelberg.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. 2013. Morfessor 2.0: Python implementation and extensions for Morfessor Baseline. Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, Department of Signal Processing and Acoustics, Aalto University.

Richard Wicentowski. 2004. Multilingual noise-robust supervised morphological analysis using the WordFrame model. In *Proc. SIGPHON*, pages 70–77. Association for Computational Linguistics.

Publication IX

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. Transfer learning and subword sampling for asymmetric-resource one-to-many neural translation. Accepted for publication in *Machine Translation*, Vol. 34, Oct 2020.

Transfer learning and subword sampling for asymmetric-resource one-to-many neural translation

Stig-Arne Grönroos · Sami Virpioja · Mikko Kurimo

Received: date / Accepted: date

Abstract There are several approaches for improving neural machine translation for low-resource languages: Monolingual data can be exploited via pretraining or data augmentation; Parallel corpora on related language pairs can be used via parameter sharing or transfer learning in multilingual models; Subword segmentation and regularization techniques can be applied to ensure high coverage of the vocabulary. We review these approaches in the context of an asymmetric-resource one-to-many translation task, in which the pair of target languages are related, with one being a very low-resource and the other a higher-resource language. We test various methods on three artificially restricted translation tasks—English to Estonian (low-resource) and Finnish (high-resource), English to Slovak and Czech, English to Danish and Swedish—and one real-world task, Norwegian to North Sámi and Finnish. The experiments show positive effects especially for scheduled multi-task learning, denoising autoencoder, and subword sampling.

Keywords Low-resource languages · Multilingual machine translation · Transfer learning · Multi-task learning · Denoising sequence autoencoder · Subword segmentation

1 Introduction

Machine translation (MT) has become an important application for natural language processing (NLP), enabling increased access to the wealth of digital information collected on-line, and new business opportunities in multilingual markets. MT has made rapid advances following the adoption of deep neural networks in the last decade, with variants of the sequence-to-sequence (seq2seq, Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014) architecture currently holding the state of the art in neural machine translation (NMT). However, the recent success has not applied to all languages equally. Current state-of-the-art methods require very large amounts of data: Seq2seq methods have been shown to work well in large data scenarios, but are less effective for low-resource languages. The rapid digitalization of society has increased the availability of suitable parallel training corpora, but the growth has not distributed evenly across languages.

The amount of data needed to reach acceptable quality can also vary based on language characteristics. Rich, productive morphology leads to a combinatorial explosion in the number of word forms. Therefore, a

Stig-Arne Grönroos
Aalto University, Department of Signal Processing and Acoustics, Espoo, Finland
Tel.: +358-40-7398282
E-mail: stig-arne.gronroos@aalto.fi
ORCiD: 0000-0002-3750-6924

Sami Virpioja
University of Helsinki, Department of Digital Humanities, Helsinki, Finland
and Utopia Analytics, Helsinki, Finland
E-mail: sami.virpioja@helsinki.fi
ORCiD: 0000-0002-3568-150X

Mikko Kurimo
Aalto University, Department of Signal Processing and Acoustics, Espoo, Finland
E-mail: mikko.kurimo@aalto.fi

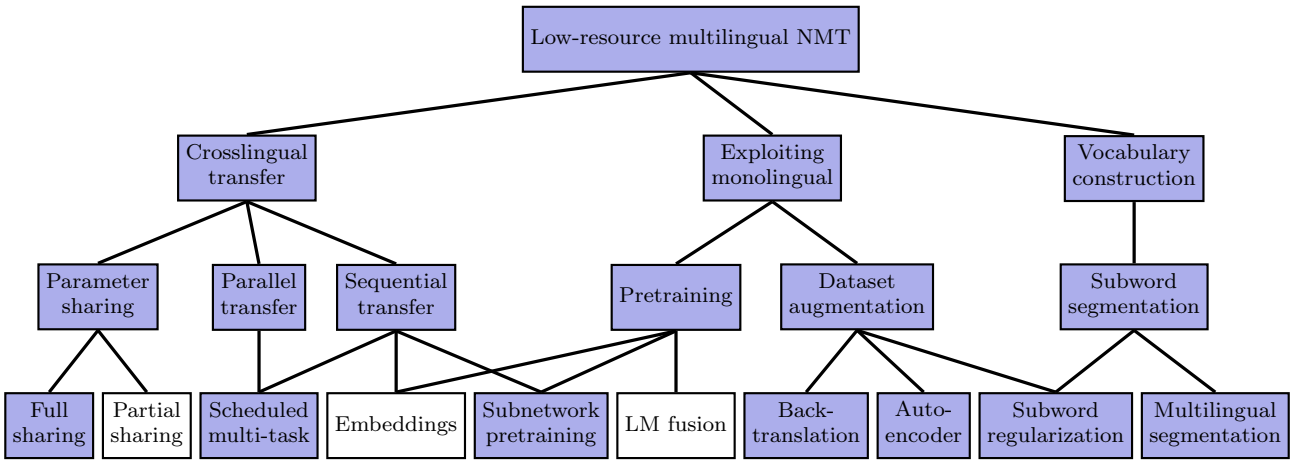


Fig. 1: Overview of techniques for improving low-resource multilingual NMT. Techniques highlighted with blue are used in this work.

larger corpus is required to reach the same coverage of word forms. Often the two challenges coincide, with morphologically complex languages that are also relatively low on resources.

Three distinct types of resources may be available for MT training: parallel data, monolingual data, and data in related languages. In the low-resource translation setting, it is primarily the parallel data that is scarce. Monolingual data is easier to acquire and typically more abundant. In addition, there may be related languages with much more abundant resources.

In this work, we consider machine translation *into* a low-resource morphologically rich language by means of ***transfer learning*** from a related high-resource target language, by exploiting available ***monolingual corpora***, and by exploring the methods and parameters for ***vocabulary construction***. Figure 1 illustrates an overview of the known techniques for low-resource multilingual NMT; most of them are considered in our experiments.

Our task is a ***one-to-many*** setting in multilingual neural machine translation (MNMT), as opposed to ***many-to-one*** and ***many-to-many*** settings (Luong, 2016). As we consider target languages that have different amounts of training resources available, we call this an ***asymmetric-resource one-to-many*** translation task. It has three major challenges:

Sparsity. Translating into a low-resource is challenging, especially in the case of a morphologically rich language, due to a combination of small data and a large target vocabulary. The resulting data sparsity makes it difficult to estimate statistics for all but the most frequent items. Even though continuous-space representations allow neural methods to generalize well, they learn poorly from low-count events. Methods like subword segmentation (Virpioja et al., 2007; Sennrich et al., 2015) can reshape the frequency distribution of the basic units to reduce sparsity, and yield a more balanced class distribution in the generator. Suitable subwords are also beneficial for exploiting transfer from related high-resource languages (Grönroos et al., 2018), and from monolingual data.

Data imbalance. In multilingual machine translation, it is very common to have an imbalance between the languages in the training data. The data can vary in quantity, quality and appropriateness of domain. Typically all three challenges affect the low-resource languages: when data is hard to come by, even noisy and out-of-domain data must be used. The data imbalance is typically addressed by oversampling the low-resource data. One way to choose the oversampling weights is using a temperature-based approach to interpolate between sampling from the true distribution and sampling uniformly (Arivazhagan et al., 2019). An alternative to oversampling the data is to adapt the gradient scale or learning rate individually for each task (Chen et al., 2018).

Task imbalance. An NMT system is a conditional language model. The training signal for the language model is much stronger than for conditioning on the source. The conditioning requires training the natural language understanding encoder and the cross-lingually aligning attention mechanism, which are both difficult tasks. High fluency is a known property of NMT (Toral and Sánchez-Cartagena, 2017; Koponen et al., 2019). When a vanilla NMT system is trained in a low-resource setting, the learning signal may be sufficient to train the language model, but insufficient for the conditioning (Östling and Tiedemann, 2017). In this case, the MT system degenerates into a fancy language model, with the output resembling generated nonsense, with

Table 1: Example from NMT system overfitted to the language modeling task.

Estonian	Source	Laktoosi puhul see nii ju ongi!
English	Overfit translation	I've been thinking about it.
English	Reference	That's the case with lactose!

possibly high fluency but little relation to the source text. As an example, Table 1 shows an output from an Estonian–English translation system trained from parallel data of only 18k sentence pairs. Mueller et al. (2020) observe this language model overfitting phenomenon in a massively multilingual but low-resource setting using Bible translations as the corpus.

Given these challenges, our research questions include:

1. On cross-lingual transfer, is it better to use sequential (pretraining followed by fine-tuning) or parallel (all tasks at the same time) transfer, or something in between?
2. On exploiting monolingual data:
   (a) For which languages should one add monolingual auxiliary tasks? Is it useful to have a target-language autoencoder in addition to the back-translation strategy, where synthetic training data is generated by a target-to-source translation model?
   (b) What kind of noise models are most useful for the denoising sequence autoencoder task?
3. On vocabulary construction:
   (a) What is a suitable granularity of subword segmentation for the low-resource task?
   (b) Does it matter what data-driven segmentation method is used?
   (c) Does subword regularization (sampling different segmentations for the same word forms) help?
4. On available data and languages:
   (a) When data is very scarce, is it better to train a small model on the low-resource data, or a larger model using also the auxiliary data?
   (b) Is cross-lingual transfer more useful than transfer from monolingual tasks?
   (c) How does the amount of the data available for the low-resource language affect the translation quality?
   (d) How important is language relatedness for the cross-lingual transfer?

As methodological contributions for NMT, we formulate a scheduled multi-task learning technique for asymmetric-resource cross-lingual transfer, propose our recently introduced Morfessor EM+Prune method (Grönroos et al., 2020) for learning the subword vocabulary, and introduce a taboo sampling task for improving the modeling of segmentation ambiguity. We include experiments using three diverse language families, with Estonian, Slovak and Danish as simulated low-resource target languages. We also contribute a Norwegian bokmål to North Sámi translation system, the first NMT system for this target language, to the best of our knowledge.

In the next three sections, we will discuss the different techniques for cross-lingual transfer, exploiting monolingual data, and vocabulary construction. Then we will describe our experimental setup and discuss the results for four different groups of languages, and finally summarize our findings.

2 Cross-lingual transfer

Multilingual training allows exploiting cross-lingual transfer between related languages by training a single model to translate between multiple language pairs. This is a form of ***multi-task learning*** (Caruana, 1998), in which each language pair in the training data can be seen as a separate learning task (Luong et al., 2015). The low-resource language is the main task, and at least one related high-resource language is used as an auxiliary task. The cardinality of the multilingual translation has an effect: cross-lingual transfer is easier in the many-to-one setting compared to one-to-many (Arivazhagan et al., 2019). For a general survey on multilingual translation, see (Dabre et al., 2020).

2.1 Sequential and parallel transfer

In transfer learning, knowledge gained while learning one task is transferred to another. The tasks can either be trained sequentially or in parallel. Transfer is essential in ***asymmetric-resource*** settings, in which the amount of training examples for the target task very small, requiring the learner to rapidly generalize. ***Sequential***

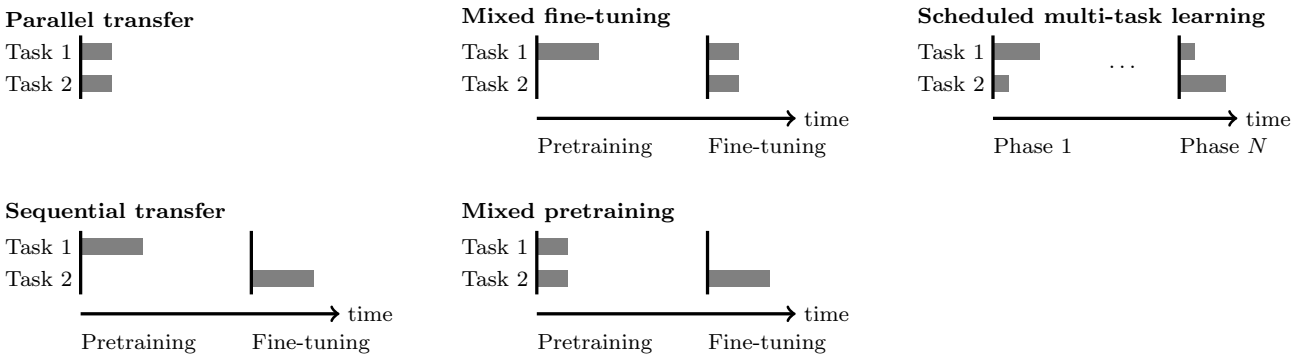


Fig. 2: Task mixing strategies for transfer learning.

transfer is a form of adaptation. In sequential transfer learning, the ***pretraining*** on a high-resource parent task is used to initialize and constrain the ***fine-tuning*** training on the low-resource child task. Zoph et al. (2016) apply sequential transfer learning to low-resource neural machine translation. Sequential transfer carries the risk of catastrophic forgetting (McCloskey and Cohen, 1989; Goodfellow et al., 2014), in which the knowledge gained from the first task fades away completely. Some parameters can be frozen between the two training phases. This reduces the number of parameters trained from the small data, which may delay overfitting.

When training tasks in parallel, called ***multi-task learning***, catastrophic forgetting does not occur. If the amount of data for different tasks is highly asymmetrical, careful tuning of the task mixture weights is critical to avoid overfitting on the small task. Sequential transfer does not require the same tuning, as convergence can be determined for each task separately.

It is also possible to combine sequential and parallel transfer. Figure 2 shows some possible ways of achieving this by mixing the tasks. One strategy—***mixed fine-tuning***—involves first pretraining only on the large task, and then fine-tuning with a mixture of tasks. Chu et al. (2017) apply this strategy to domain adaptation. Kocmi (2019) try the inverse setting—***mixed pretraining***—pretraining on a mixture of tasks and fine-tuning only on the child task.

Kiperwasser and Ballesteros (2018) propose generalizing these strategies into ***scheduled multi-task learning***, in which training examples from different tasks are selected according to a mixing distribution. The mixing distribution changes during training according to the task-mix schedule. They experiment with three schedules: constant, exponential and sigmoidal. We propose a new partwise constant task-mix schedule suitable for an asymmetric-resource setting with multiple auxiliary tasks. The task-mix schedule can have an arbitrary number of steps, any of which can be mixing multiple tasks. All the other strategies can be recovered by using particular schedules with scheduled multi-task learning.

2.2 Parameter sharing

In neural networks, multilingual models are implemented through parameter sharing. It is possible to share all neural network parameters, or select a subset for sharing allowing the remaining ones to be language-specific. Parameter sharing can be either hard or soft. In hard parameter sharing the exact same parameter matrix is used for several languages. In soft parameter sharing, each language has its own parameter matrix, but a dependence is constructed between the corresponding parameters for different languages.

The ***target language token*** Johnson et al. (2017) and ***language embedding*** (Conneau and Lample, 2019) approaches use hard sharing of all parameters. In the former, the model architecture is the same as in a language-pair-specific model. The target language is indicated by a preprocessing step that prepends to the input a special target language token, e.g. ⟨TO_FI⟩ to indicate that the target language is Finnish. The approach can be scaled to more languages by increasing the capacity of the model, primarily by increasing the depth in layers (Arivazhagan et al., 2019). The latter can be described as a factored representation, with the language embedding factor marking the language of each word on the target side.

In contrast to full parameter sharing, it is also possible to divide the model parameters into shared and ***language-specific subnetworks***, e.g. sharing all parameters of the encoder, while letting each target language have its own decoder. Parameter sharing can even be controlled on a more fine-grained level (Sachan and Neubig,

2018). Shared attention (Firat et al., 2016) uses language-specific encoders and decoders with a shared attention, while language-specific attention (Blackwood et al., 2018) does the opposite by sharing only the feedforward sublayers of the decoder, while using language-specific parameters for the attention mechanisms.

The ***contextual parameter generator*** (Platanios et al., 2018) meta-learns a soft dependency between parameters for different tasks. It does this by using one neural network (the parameter generator) to generate from some contextual variables the weights of another network (the model). Gu et al. (2018) apply meta-learning to find initializations that can very rapidly adapt to a new low-resource source language.

3 Exploiting monolingual data

While parallel data is the primary type of data used for training MT models, methods for effectively exploiting the more abundant monolingual data can greatly increase the number of available examples to learn from. Use of monolingual data can be viewed as semi-supervised learning: both labeled (parallel) and unlabeled (monolingual) data are used. There are two main approaches to exploiting monolingual data in MT: transfer learning and dataset augmentation.

3.1 Transfer learning: monolingual pretraining

In monolingual pretraining, some of the parameters of the final translation model are pretrained on a task using monolingual data, possibly using a different loss than the one used during NMT training. There are several ways to use pretraining: Pretrain word (or subword) ***embeddings*** for the encoder, decoder, or both. Pretrain a separate ***language model*** for the target language, and combine it with the predictions of the translation model. Or, finally, pretrain an entire ***subnetwork***—encoder or decoder—of the translation model.

3.1.1 Embeddings

Source and target embeddings can be pretrained on monolingual data from the source and target languages, respectively (Di Gangi and Federico, 2017). Alternatively, joint cross-lingual embeddings can be trained on both (Artetxe et al., 2018). As the embeddings are trained for e.g. a generic contextual prediction task, this is a form of transfer learning. The pretrained embeddings can either be frozen or fine-tuned, by respectively omitting or including them as trainable parameters during NMT training. Thompson et al. (2018) investigate the effects of freezing various subnetwork parameters—including embeddings—on domain adaptation. In addition to using monolingual data, pretrained embeddings can contribute to cross-lingual transfer in the case of a shared multilingual embedding space (Artetxe et al., 2018). The shared embedding spaces are typically on a word level.

3.1.2 Language model fusion

The predictions of a strong language model can be combined with the predictions of the translation model, either using a separate rescoring step, or by combining the predictions during decoding, using ***model fusion***. This approach is used in statistical machine translation, where one or more target language models are combined with a statistical translation model. The approach can also be applied in neural machine translation, through shallow fusion, deep fusion (Gulcehre et al., 2015), cold fusion (Sriram et al., 2017), or PostNorm (Stahlberg et al., 2018). As a neural machine translation system is already a conditional language model, it may be preferable to find a way to train the parameters of the NMT system using the monolingual data.

3.1.3 Subnetwork pretraining

In subnetwork pretraining, the intent is to pretrain entire network components—the encoder or the decoder—with knowledge about the structure of language. One way to achieve this using unlabeled data is to apply a language modeling loss during pretraining. The loss function can either be the traditional next token prediction, or a masked language model. Alternatively an autoencoder loss can be used.

Domhan and Hieber (2017) modify the NMT architecture by adding an auxiliary language model loss in the internal layers of the decoder, before attending to the source. This loss allows the first layers of the decoder to be trained on monolingual data. They find no benefit of adding the language model loss unless additional monolingual

data is used. Adding monolingual data gives a benefit, but does not outperform back-translation. Ramachandran et al. (2017) pretrain the encoder and decoder with source and target language modeling tasks, respectively. To prevent overfitting, they use task-mix fine-tuning: the translation and language modeling objectives are trained jointly (with equally weighted tasks). Skorokhodov et al. (2018) use both pretraining (on both source and target side) and gated shallow fusion (on the target side) to transfer knowledge from pretrained language models. Some of the experiments are performed on low-resource data going down to 10k sentence pairs.

3.2 Dataset augmentation

The easiest way to improve generalization is to train on more data. As natural training data is limited, a practical way to acquire more is to generate additional synthetic data for augmentation. The main benefit of dataset augmentation is as regularization to prevent overfitting to non-robust properties of small data.

Simple ways to generate synthetic data include using a single dummy token on the source side (Sennrich et al., 2016), and copying the target to source (Currey et al., 2017). The latter can be interpreted as a target-side autoencoder task without noise. The largest factor in determining the effectiveness of using synthetic data is how much the synthetic data deviates from the true data distribution. To avoid confusing the encoder with synthetic data from a different distribution than the natural data, it may be beneficial to use a special tag to identify the synthetic data (Caswell et al., 2019).

3.2.1 Back-translation

Synthetic data can be self-generated by the model being trained, or a related model. In machine translation, the best known example of synthetic data is ***back-translation*** (BT) (Sennrich et al., 2016). The process of back-translation begins with the training of a preliminary MT model in the reverse direction, from target to source. The target language monolingual data is translated using this model, producing a synthetic, pseudo-parallel data set with the potentially noisy MT output on the source side. Because the quality of the translation system used for the back-translation affects the noisiness of the synthetic data, the procedure can be improved by iterating with alternating translation direction (Lample et al., 2018b). Edunov et al. (2018) propose adding noise to the back-translation output. The benefit of noisy back-translation is further analyzed by Graça et al. (2019), who recommend turning off label smoothing in the reverse model when combined with sampling decoding. As a related strategy, Karakanta et al. (2018) convert parallel data from a high-resource language pair into synthetic data for a related low-resource pair using transliteration. Zhang and Zong (2016) exploit monolingual data in two ways: through self-learning by "forward-translating" the monolingual source data to create synthetic parallel data, and by applying a reordering auxiliary task: the input is the natural source text, while the output is the source text reordered using rules to match the target word order.

3.2.2 Subword regularization

Subword regularization is a technique proposed by Kudo (2018) for applying a probabilistic subword segmentation model to generate more variability in the input text. Each time a word token is used during training, a new segmentation is sampled for it. It can be seen as treating the subword segmentation as a latent variable. While marginalizing over the latent variable exactly is intractable, the subword regularization procedure approximates it through sampling.

3.2.3 Denoising sequence autoencoder

Back-translation is a slow method due to the additional training of the reverse translation model. A computationally cheaper way to turn monolingual data into synthetic parallel data is to use a denoising autoencoder as an auxiliary task. Target language text, corrupted by a noise model, is fed in as a pseudo-source. Different noise models can be used, e.g. applying reordering, deletions, or substitutions to the input tokens. The desired reconstruction output is the original noise-free target language text.

An autoencoder (Bourlard and Kamp, 1988) is a neural network that is trained to copy its input to its output. It applies an encoder mapping from input to a hidden representation, i.e. code $\boldsymbol{h} = f(\boldsymbol{x})$, and decoder mapping from code to a reconstruction of the input $\hat{\boldsymbol{x}} = g(\boldsymbol{h})$. To force the autoencoder to extract patterns in the data instead of finding the trivial identity function $\hat{\boldsymbol{x}} = \mathbf{1}(\mathbf{1}(\boldsymbol{x}))$, the capacity of the code must be restricted somehow.

In the undercomplete autoencoder, the restriction is in the form of a bottleneck layer with small dimension. For example, in the original sequence autoencoder (Dai and Le, 2015), the entire sequence is compressed into a single vector.

In a modern sequence-to-sequence architecture, the attention mechanism ensures a very large bandwidth between encoder and decoder. When used as an autoencoder, the network is thus highly overcomplete. In this case, the capacity of the code has to be controlled by regularization. Robustness to noise is used as the regularizer in the ***denoising autoencoder*** (Vincent et al., 2008). Instead of feeding in the clean example $\boldsymbol{x}$, a corrupted copy of the input is sampled from a noise model $\mathrm{C}(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$. The denoising autoencoder must then learn to reverse the corruption to reconstruct the clean example. The use of noise as regularization is a successful technique used e.g. in Dropout (Srivastava et al., 2014), label smoothing (Szegedy et al., 2016), and SwitchOut (Wang et al., 2018). Also multi-task learning acts as regularization by claiming some of the capacity of the model. Belinkov and Bisk (2017) apply both natural and synthetic noises for NMT evaluation, finding that standard character-based NMT models are not robust to these types of noise.

There are multiple ways of adding the autoencoder loss to the NMT training. The simplest one treats the autoencoder task as if it was another language pair for multilingual training, and involves no changes to the architecture. When using this type of autoencoder task on target language sentences, the task cardinality changes into a many-to-one problem: the model must simultaneously learn a mapping from source to target and from corrupted target to clean target. In both tasks the target language is the same. As the decoder is a conditional language model, this task strengthens the modeling of the target language. When using source language sentences, the model must simultaneously learn a one-to-many mapping from source to target and from corrupted source to clean source. Thus the decoder must learn to output both languages. The task may strengthen the encoder, by increasing its robustness to noise, and by preventing the encoding from becoming too specific to the target language. Luong et al. (2015) and Luong (2016) experiment with various auxiliary tasks, including this type of autoencoder setup. They see a benefit of using the autoencoder task, as long as it has a low enough weight in the task mix. This setup is used also in our experiments.

There are also more complex NMT autoencoder setups. In ***dual learning***, the autoencoder is built from source-to-target and target-to-source translation models. He et al. (2016) combine source-to-target and target-to-source translations in a closed loop which can be trained jointly, using two additional language modeling tasks (for source and target respectively), and reinforcement learning with policy gradient. Cheng et al. (2016) use a dual learning setup to exploit monolingual corpora in both source and target languages. Their loss consists of four parts: translation likelihoods in both directions, source autoencoder, and target autoencoder. Tu et al. (2017) simplify the dual learning setup into an encoder–decoder–reconstructor network. The reconstructor attends to the final hidden states of the decoder and thus does not need a separate encoder. Their aim is to improve adequacy by penalizing undertranslation: the reconstructor is not able to generate any parts of the sentence omitted by the decoder.

3.2.4 Noise models for text

To apply a denoising autoencoder to text, a suitable noise model for text is needed. In domains such as image and speech, there are very intuitive noises, including rotating, scaling, and mirroring for images; and reverberation, time-scale stretching, and pitch shifting for speech. As text is a sequence of discrete symbols, where even a small change can have a drastic effect on meaning, suitable noise models are less intuitive. It is not feasible to guarantee the noise does not change the correct translation of the input.

Local reordering. Lample et al. (2018a) perform a local reordering operation σ that they call ***slightly shuffling*** the sentence. The reordering is achieved by adding to the index i of each token a random offset drawn from the uniform distribution from 0 to a maximum distance k. The tokens are then sorted according to the offset indices. This maintains the condition $\forall i \in \{1, n\}, |\sigma(i) - i| \leq k$.

Token deletion. Randomly dropping tokens is perhaps the most commonly used noise. It is the central idea in ***word dropout*** (Iyyer et al., 2015). In word dropout, each token is dropped according to a Bernoulli distribution parameterized by a tunable dropout probability.

Token insertion. Randomly selected tokens can also be inserted into the sentence. The tokens can be sampled from the entire vocabulary, or from a particular class of tokens. E.g. Vaibhav et al. (2019) insert three classes of tokens: stop words, expletives, and emoticons.

Token substitution. SwitchOut (Wang et al., 2018) applies random substitutions to tokens both in the source and the target sentence. One benefit of SwitchOut is that it can easily and efficiently be applied late in the data processing pipeline, even to a numericalized and padded minibatch. Any noises that affect the length of the sequence are best applied before numericalization.

Token masking. Masked language models (Devlin et al., 2019; Song et al., 2019; Lewis et al., 2019; Joshi et al., 2020) apply a special case of token substitution, randomly substituting tokens or spans of tokens with a mask symbol.

Word boundary noise. In a special case of token substitution, the substituted token is selected deterministically as the token with a word boundary marker either added or removed. E.g. *"kielinen"* would be substituted by *"␣kielinen"* and vice versa. This might improve robustness to compounding mistakes such as *"*suomen kielinen"* (Finnish speaker).

Taboo sampling. In addition to training the translation model, the idea of subword regularization (Kudo, 2018) can be used in the autoencoder. Here, we propose taboo sampling as a special form of subword regularization for monolingual data. The method takes a single word sequence as input, and outputs two different segmentations for it. The two segmentations consist of different subwords, whenever possible. Only single character morphs are allowed to be reused on the other side, to avoid failure if no alternative exists. E.g. *"unreasonable"* could be segmented into *"un ++reasonable"* on the source side and *"unreason ++able"* on the target side. When converted into numerical indices into the lexicon, these two representations are completely different. The task aims to teach the model to associate with each other the multiple ambiguous ways to segment a word, by using a segmentation-invariant internal representation.

For each word, one segmentation is sampled in the usual way, after which another segmentation is sampled using taboo sampling. During taboo sampling, all multi-character subwords used in the first segmentation have their emission probability temporarily set to zero. To avoid introducing a bias from having all the taboo sampled segmentations on the same side, the sides are mixed by uniformly sampling a binary mask of the same length as the sentence from the set of masks with half the bits 1. All words for which the mask bit is set have the source and target segmentations swapped.

Proposed noise model combinations. Our proposed noise model combination is depicted in Figure 3. It consists of three pipelines: The pipeline for parallel data (a) consists of only sampling segmentation. The primary pipeline for monolingual data (b) is a concatenation of multiple noise models: local reordering, segmentation, and token deletion. A secondary pipeline for monolingual data (c) uses taboo segmentation. In all cases the output consists of a pair of source and target sequences.

Observe that the transformations are applied in the data loader at training time, not as an off-line preprocessing stage. This allows the noise to be resampled for each parameter update, which is critical when training continues for multiple epochs of a small dataset. As a minor downside, the NMT software needs to be modified to accommodate the heavier data loader, while preprocessing generally requires no modifications to the software.

4 Vocabulary construction

The vocabulary or lexicon of a translation model is the set of basic units or building blocks the text is decomposed into. In phrase-based machine translation, the standard approach is to use a word lexicon. Segmentation into subword units has been proposed mostly for morphologically rich languages, for which a word lexicon leads to very high out-of-vocabulary (OOV) rates (Lee, 2004; Oflazer and El-Kahlout, 2007; Virpioja et al., 2007), and character segmentation for closely related languages (Tiedemann, 2009). However, the change of paradigm to neural machine translation has changed also the practice in vocabulary construction: With the exception of unsupervised translation based on pretrained word embeddings (Artetxe et al., 2018; Yang et al., 2018), the standard approach for models is segmentation into subword units (Sennrich et al., 2015). Some studies aim even to the other extreme, characters (Chung et al., 2016; Costa-jussà and Fonollosa, 2016) or bytes (Costa-jussà et al., 2017).

A specific task in subword segmentation is the morphological surface segmentation. There the aim is to split words into morphs, the surface forms of meaning-bearing sub-word units, morphemes. The concatenation of the morphs is the word, for example

$$capability \mapsto cap +\!\!+ abil +\!\!+ ity.$$

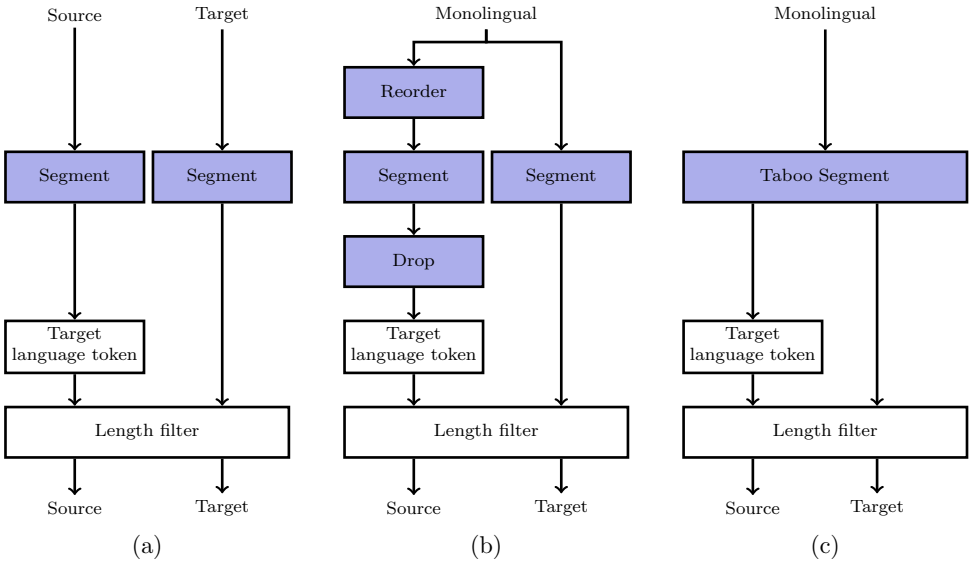


Fig. 3: Transformations applied to data at training time. Steps with blue background are part of the stochastic noise model. Steps with white background are the deterministic target language token prefixing and length filtering. Length filtering must be applied after segmentation, which may make the sequence longer.

Unsupervised morphological segmentation, dating back to Harris (1955), was an active research topic in 2000s and early 2010s (Goldsmith, 2001; Creutz and Lagus, 2007; Hammarström and Borin, 2011), and the methods have been evaluated in various NLP applications (Kurimo et al., 2010; Virpioja et al., 2011). However, in applications based on neural network models, such as NMT, the correspondence of the subwords to linguistic morphemes is not of high importance, as the encoders are able to determine the meaning of the units in context. Therefore the subword segmentation is typically tuned using other criteria, such as the size of subword lexicon or the frequency distribution of the units. Desirable characteristics for a vocabulary to be used in multilingual NMT include:

1. **high coverage** of the training data, without imbalance between languages,
2. **a tractable size** for training, and
3. the right **level of granularity** for cross-lingual transfer.

Without a high coverage, some parts of the training data are impossible to represent using the vocabulary. The unrepresentable parts may be replaced with a special "unknown" token. If the proportion of unknown tokens increases, translation quality deteriorates. In a multilingual setting, a common approach is to use a shared subword vocabulary between the multiple source or target languages. In this case, training the segmentation model with a balanced data distribution is important to provide high coverage also for the less resourced languages.

Vocabulary size affects both the memory complexity via the number of network parameters and the computational cost via the length of the sequences and the size of the softmax layer. When using large vocabularies, e.g. words, the sequences are short, but vocabularies may grow intractably large, particularly for morphologically complex languages. When using small vocabularies, e.g. characters, memory requirements are low, but long sequences make training slow, particularly for recurrent networks.

The granularity of the segmentation affects both coverage and size of the lexicon: finer granularity typically means better coverage and smaller lexicon size. However, within the reasonable limits set by the coverage and size, it is much harder to determine the best possible level of granularity. Recent research (Cherry et al., 2018; Kreutzer and Sokolov, 2018; Arivazhagan et al., 2019) indicates that smaller subwords are particularly useful for cross-lingual transfer to low-resource languages in supervised settings. Exploiting similarity of related languages by increasing the consistency of the segmentation between similar words of the source and target language can also be useful (Grönroos et al., 2018). In unsupervised NMT (Artetxe et al., 2018), cross-lingual transfer requires basic units to be aligned between languages without use of parallel data. When starting with pretrained embeddings, longer units are typically used, as they carry more meaning than short units. It is therefore an open question how the optimal segmentation granularity varies with the amount of resources available.

Next, we consider different data-driven segmentation methods proposed for machine translation. This study focuses on segmentation methods applying a ***unigram language model***. In the unigram language model, it is assumed that the morphs in a word occur independently of each other. Given the parameters $\boldsymbol{\theta}$ of the segmentation model, the probability of a sequence of morphs $\boldsymbol{s}$ decomposes into the product of the probabilities of the morphs m of which it consists:

$$P_{\boldsymbol{\theta}}(\boldsymbol{s}) = \prod_{i=1}^{N} P_{\boldsymbol{\theta}}(m_i), \tag{1}$$

4.1 Byte Pair Encoding

The most popular method for subword segmentation in the field of NMT is currently the Byte Pair Encoding (BPE) compression algorithm (Gage, 1994). The BPE algorithm iteratively replaces the most frequent pair of bytes in the data with a single unused byte. In NMT, the algorithm is typically used on characters, and the merging of characters is stopped when the given vocabulary size is reached (Sennrich et al., 2015). While BPE is not a probabilistic model, the coding resembles unigram language models in that every subword m_i is encoded individually. As a bottom-up algorithm, BPE is reasonable to use in multilingual settings just by concatenating the corpora before training; this approach is called ***joint*** segmentation (Sennrich et al., 2015). If the data is balanced over the languages, the frequent words will be constructed in the early steps of the algorithm for all languages.

4.2 SentencePiece

SentencePiece (Kudo, 2018; Kudo and Richardson, 2018) is another segmentation method proposed especially for NMT. In contrast to BPE, it defines a proper statistical model for the unigram model in Equation 1, and tries to find the model parameters that maximize likelihood of the data given a constraint on the vocabulary size.

For training the model, SentencePiece applies the Expectation Maximization (EM) algorithm (Dempster et al., 1977). The EM algorithm only updates the expected frequencies of the current units; it is not able to add or remove subwords from the vocabulary. Thus to use EM for the segmentation problem, two other things are needed: a ***seed lexicon*** and a ***pruning phase***. The seed lexicon initializes the vocabulary with useful candidate units, and pruning phase removes the least probable units from the model. Prior to SentencePiece, a similar approach has been proposed by Varjokallio et al. (2013) for application in automatic speech recognition.

In SentencePiece, the seed lexicon is constructed from the most frequent substrings in the training data. After initializing the seed lexicon, SentencePiece alternates between the EM phase and the pruning phases until the desired vocabulary size is reached. In the pruning phase, the subwords are sorted by the reduction in the likelihood function if the subword was removed. A certain proportion (e.g. 25%) of the multi-character subwords are pruned at a time, followed by the next EM phase.

4.3 Morfessor EM+Prune

Morfessor is a family of generative models for unsupervised morphology induction (Creutz and Lagus, 2007). Here, consider the Morfessor Baseline method (Creutz and Lagus, 2002; Virpioja et al., 2013) and its recent Morfessor EM+Prune variant (Grönroos et al., 2020).

4.3.1 Model and cost function

Morfessor Baseline is applies the unigram language model (Equation 1). In contrast to SentencePiece, Morfessor finds a point estimate for the model parameters $\hat{\boldsymbol{\theta}}$ using Maximum a Posteriori (MAP) estimation. The MAP estimate yields a two-part cost function, consisting of a prior (the lexicon cost) and likelihood (the corpus cost). The Morfessor prior, inspired by the Minimum Description Length (MDL) principle (Rissanen, 1989), favors lexicons containing fewer, shorter morphs.

For tuning the model, Kohonen et al. (2010) propose weighting the likelihood with a hyper-parameter α:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \{ -\log \overbrace{P(\boldsymbol{\theta})}^{\text{prior}} - \alpha \log \overbrace{P(\boldsymbol{D} \mid \boldsymbol{\theta})}^{\text{likelihood}} \} \tag{2}$$

This parameter controls the granularity of the segmentation. High values increase the weight of each emitted morph in the corpus (less segmentation), and low values give a relatively larger weight to a small lexicon (more segmentation).

Similar to SentencePiece, Morfessor can be used in subword regularization (Kudo, 2018). Alternative segmentations can be sampled from the full data distribution using the forward-filtering backward-sampling algorithm (Scott, 2002) or approximatively from an n-best list.

4.3.2 Training algorithm

The original training algorithm of the Morfessor Baseline method, described in more detail by Creutz and Lagus (2005) and Virpioja et al. (2013), is a local greedy search. The lexicon is initialized by whole words, and the segmentation proceeds recursively top-down, finding an optimal segmentation into two parts for the current word or subword unit. Our preliminary studies have indicated that this algorithm does not find as good local optima as the EM algorithm especially for the small lexicons useful in NMT. As a solution, we have developed a new variant of the method called Morfessor EM+Prune (Grönroos et al., 2020).[1] It supports the MAP estimation and MDL-based prior of the Baseline model, but implements a new training algorithm based on the EM algorithm and lexicon pruning inspired by SentencePiece.

The training algorithm starts with a seed lexicon and alternates the EM and lexicon pruning steps similarly to SentencePiece. The prior of the Morfessor model must be slightly modified for use with the EM algorithm, but the standard prior is used during pruning. While SentencePiece aims for a predetermined lexicon size, in Morfessor, the final lexicon size is controlled by the hyper-parameter α (Equation 2). To reach a subword lexicon of a predetermined size while using the prior, Morfessor EM+Prune implements an automatic tuning procedure. When the estimated change in prior and likelihood are computed separately for each subword, the value of α that gives exactly the desired size of lexicon after the pruning can be calculated.

In earlier work (Grönroos et al., 2020), we have shown that the EM+Prune algorithm reduces search error during training, resulting in models with lower costs for the optimization criterion. Moreover, lower costs lead to improved accuracy when segmentation output is compared to linguistic morphological segmentation. In the present study, we test it for the first time in NMT.

5 Experiments

In the experiments, we study how to best exploit the additional monolingual and cross-lingual resources for improving machine translation into low-resource morphologically rich languages. We compare various methods for three major aspects affecting the translation quality: using cross-lingual transfer, exploiting monolingual data and applying subword segmentation. The main focus lies on a noise model incorporating the subword segmentation.

We target a one-to-many multilingual setting with related, morphologically rich languages on the target side. The related languages include both high- and low-resource languages. This setting provides a good opportunity for cross-lingual learning, as the amount of data is highly asymmetric. Our aim is not to achieve an interlingual representation, so allowing the encoder to specialize for target languages is acceptable if it improves performance.

5.1 Data sets

We perform experiments on four translation tasks, each consisting of a language triple: source language (SRC), high-resource target language (HRL) and low-resource target language (LRL). We only show SRC-LRL translation results, as the goal is to improve this particular translation direction.

The four tasks (LRL in boldface) are:

[1] Software available at `https://github.com/Waino/morfessor-emprune`.

1. English (ENG) to Finnish (FIN) and **Estonian** (EST),
2. English to Czech (CZE) and **Slovak** (SLO),
3. English to Swedish (SWE) and **Danish** (DAN),
4. Norwegian bokmål (NOB) to Finnish (FIN) and **North Sámi** (SME).

In each task the two target languages are related. The target languages belong to three different language families: Germanic, Balto-Slavic and Uralic. All target languages are morphologically complex.

We use as parallel corpora Europarl (Koehn, 2005), and OpenSubtitles v2018 (Lison and Tiedemann, 2016), when available. In addition, we use the eu, news, and subtitle domains of CzEng v1.7 (Bojar et al., 2016), and the UiT freecorpus[2]. The corpora used for each language pair are shown in Table 2. The domains for the training data are parliamentary debate, movie subtitles, news and web, with the exception of North Sámi which contains a mix of many domains.

Our main source of monolingual data is WMT news text[3]. In addition, we use the following monolingual corpora: skTenTen[4] and Categorized News Corpus[5] for Slovak, Riksdagens protokoll[6] for Swedish, News 2012[7] for Danish, Aviskorpus[8] for Norwegian, and Wikipedia[9] for North Sámi.

Table 2: Parallel corpora.

		Europarl	OpenSubtitles	Other parallel
ENG	CZE			CzEng
ENG	SLO	✓	✓	
ENG	FIN	✓	✓	Rapid2016, Paracrawl
ENG	EST	✓	✓	
ENG	SWE	✓	✓	
ENG	DAN	✓	✓	
NOB	FIN		✓	
NOB	SME			UiT freecorpus

For each of the low-resource languages, we select a subset of 18k sentence pairs. For ENG-EST, we also perform an experiment where the low-resource subset is repeatedly subsampled down to 3k sentence pairs. To avoid introducing a domain imbalance in the sampled subset, the pairs are sampled such that an equal number of sentences are selected uniformly at random from each cleaned corpus. The training data sizes after cleaning and subsampling are shown in Table 3.

Table 3: Data set sizes after cleaning.

			Parallel			Monolingual		
SRC	HRL	LRL	SRC-HRL	SRC-LRL	BT	SRC	HRL	LRL
ENG	CZE	SLO	24.7M	(18k)	1M	44.3M	13.6M	27.8M
ENG	FIN	EST	19.4M	(18k)	1M	44.3M	6.3M	3.6M
ENG	SWE	DAN	11.5M	(18k)	750k	44.3M	10.7M	950k
NOB	FIN	SME	4.9M	152k	150k	40.1M	6.3M	181k

As test sets we use the WMT newstest2018 (Bojar et al., 2018) for ENG-EST, the WMT test2011 extended to Slovak by Galuščáková and Bojar (2012) for ENG-SLO. For ENG-DAN we use 2k sentence pairs sampled from the JRC-Acquis corpus (Steinberger et al., 2006). For NOB-SME we use the Apertium story "Where is James?", a 48-sentence text with simple language, used as an initial development set for Apertium rule based MT systems (Forcada et al., 2011).

[2] https://victorio.uit.no/freecorpus/
[3] http://www.statmt.org/wmt18/translation-task.html
[4] http://hdl.handle.net/11858/00-097C-0000-0001-CCDB-0
[5] Technical University of Kosice, 2014
[6] https://spraakbanken.gu.se/eng/resource/rd-prot
[7] http://hdl.handle.net/11022/0000-0000-2238-B
[8] https://www.nb.no/sprakbanken/show?serial=oai%3Anb.no%3Asbr-4&lang=en
[9] sewiki-20191201 dump

5.2 Evaluation measures

When selecting the evaluation measures, the morphologically rich target languages must be taken into account. Therefore, we use Character-F_1 (Popović, 2015) in addition to BLEU[10] (Papineni et al., 2002). To evaluate the performance of systems on rare words, we use word unigram F_1 score computed over words occurring less than 5 times in the parallel training data (Sennrich et al., 2015).

5.3 Training details

Table 4: Specifications for the NMT system.

Encoder	8 Transformer layers	Label smoothing	0.1
Decoder	8 Transformer layers	Precision	16-bit floating point
Hidden size	1024	Minibatch size	9200 subword tokens
Filter size	4096	Gradient accumulation	4 minibatches
Attention heads	16	Effective minibatch size	36800 subword tokens
Adam beta2	0.997	Training time	100k steps
Warmup	noam, 16k steps	Beam size	8
Dropout weight	0.1	Heuristic penalties	None

We use the Transformer NMT architecture (Vaswani et al., 2017). Model hyper-parameters are shown in Table 4. Training takes approximatively 96h on a single V100 GPU, with the data loader in a separate process. When using scheduled multi-task learning, the mixing distribution is changed after 40k steps. In all experiments, we apply full parameter sharing using a target language token. We tune our models towards the best product of the three evaluation measures (charF_1, BLEU, rare word F_1) on a development set.

Back-translation was performed with essentially the same system, but with sources and targets swapped to achieve a many-to-one configuration. We mark the back-translation data as synthetic using a special token.

When using subword regularization or denoising autoencoder, the training data is not simply loaded from disk, but new random segmentations and noises are sampled each time a training example is used. To alleviate slowdown, we moved the dataloader and preprocessing pipeline into a separate process, which communicates the numericalized and padded minibatches to the training process via a multiprocessing queue. Our data loader is implemented as a fork of OpenNMT-py[11] (Klein et al., 2017).

With multilingual training, autoencoders and back-translation, our setting involves a large number of different tasks. The tasks can be divided by language (HRL, LRL) and by type (translation, autoencoder). Nearly all runs, with the exception of our vanilla baseline, use a mix of tasks in some or all phases.

5.4 Results

In this section, we present the results of ten experiments, each exploring a separate aspect of asymmetric-resource one-to-many NMT. We have detailed results for English–Estonian, and verify the central findings on two additional language triples. Finally, we present some results on the actual low-resource pair Norwegian–North Sámi.

Unless otherwise stated, the compared models are trained using joint Morfessor EM+Prune segmentation with 16k subword vocabulary, cross-lingual scheduled multi-task learning, autoencoder with full noise model, and subword regularization for the translation task. Our initial results are using autoencoder tasks for all three languages (SRC+HRL+LRL). Later some of the results were rerun with the better SRC+LRL configuration, which omits the high-resource target language autoencoder.

[10] mteval-v13a.pl

[11] Software available at `https://github.com/Waino/OpenNMT-py/tree/dynamicdata`. Later, the dataloader of OpenNMT-py version 2.0 was redesigned to incorporate our proposals.

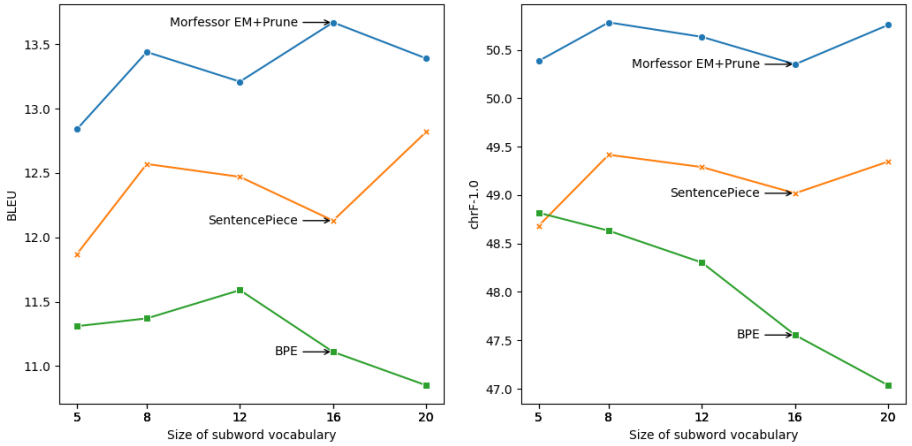


Fig. 4: Varying the subword vocabulary. Multilingual models, with SRC+HRL+LRL autoencoder and full noise model, except for BPE which are multilingual models without autoencoder or noise. Results on English→Estonian newsdev2018.

5.4.1 Subword segmentation

For subword segmentation, we compare Morfessor EM+Prune to SentencePiece on various vocabulary sizes. The results are shown in Figure 4. There is no clear optimal vocabulary size: in particular for the Character F_1 measure the performance remains nearly constant. On the test set, Morfessor EM+Prune is +0.6 BLEU better than SentencePiece. The difference is smaller than the +1.48 BLEU difference on the development set, but consistent. The difference between Morfessor EM+Prune and SentencePiece is similar for the ENG-DAN and ENG-SLO translation directions. In preliminary experiments BPE gave 0.65 BLEU worse results than EM+Prune already without subword regularization. We decided against further experiments using BPE, as it is incompatible with subword regularization.

5.4.2 Cross-lingual transfer

Table 5 shows the effect of multilingual training, with and without the autoencoder task. The cross-lingual transfer from the high-resource language yields the largest single improvement in our experiments. The multilingual model without autoencoder performs between +10.26 and +12.7 BLEU better than the vanilla model using only LRL parallel data. Adding an autoencoder loss results in a smaller gain, between +4.97 and +5.55 BLEU. The gains are partly cumulative for an additional gain of +0.05 to +1.14 BLEU.

Table 5: Results for cross-lingual transfer. Abbreviations: ML for multilingual, BT for back-translation, AE for autoencoder.

			Autoencoder			ENG–EST			ENG–DAN			ENG–SLO		
Method	ML	BT	SRC	HRL	LRL	chrF1	BLEU	rare	chrF1	BLEU	rare	chrF1	BLEU	rare
Both	✓		✓		✓	51.71	14.04	34.79	50.06	13.92	54.58	50.19	14.02	69.94
Only ML	✓					50.09	12.90	33.20	49.57	13.13	54.21	49.83	13.97	68.79
Only AE			✓		✓	42.65	8.19	21.59	42.26	7.60	44.48	38.97	6.25	62.51
Neither (vanilla)						29.46	2.64	6.22	31.95	2.63	30.40	23.76	1.27	36.80

The results for the vanilla model use a smaller configuration, with 4 encoder and 4 decoder layers, and batch size reduced to 2048. For the vanilla model the small network performed better than the large one, but when adding either multilingual training or autoencoder, the large network is superior.

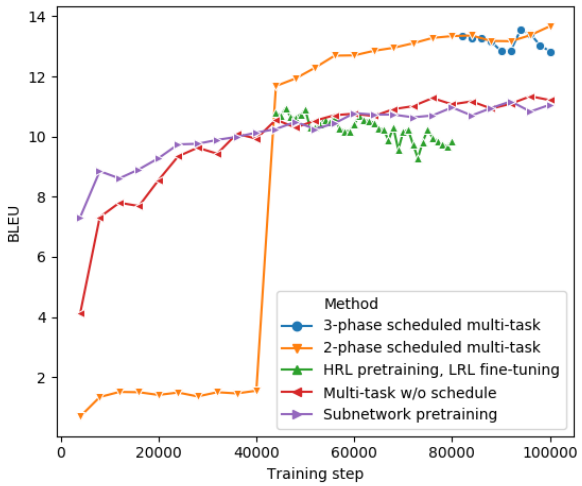


Fig. 5: Learning curves on LRL English→Estonian development set. Multilingual models, with SRC+HRL+LRL autoencoder and full noise model. Note that up to 40k training steps, the model using scheduled multi-task learning has not seen any LRL data.

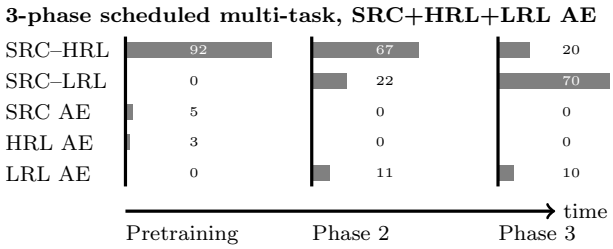


Fig. 6: The task mix schedule used in the 3-phase scheduled multi-task learning experiment. The 2-phase schedule is the same, except it omits the third phase, continuing the second phase until the end of training.

5.4.3 Scheduled multi-task learning

Figure 5 shows the learning curves on the development set and Table 6 the evaluations on the test set for different configurations of transfer learning.

Multi-task without schedule is trained with a constant task mixing distribution. The result marked *HRL pretraining, LRL fine-tuning* uses a mix of HRL translation and autoencoder tasks for pretraining, and only a single task—LRL translation—for fine-tuning, and is thus fully sequential in terms of languages. It quickly overfits in the fine-tuning phase.

The models using scheduled multi-task learning combine sequential and parallel transfer. In *2-phase scheduled multi-task*, LRL tasks are not used in the pretraining phase, but a mix of tasks is used for fine-tuning. It gives a benefit of +2.4 BLEU compared to the model fine-tuning on only LRL tasks, and +1.77 BLEU compared to training with a constant mixing distribution. The *3-phase scheduled multi-task* adds a third phase training mostly on LRL tasks. A small proportion of HRL translation is included to delay overfitting. The model again overfits in the final phase, but does reach a higher score before doing so. The 3-phase task mixing schedule is shown in Figure 6.

Table 6: Results for scheduled multi-task learning.

			Autoencoder			ENG–EST			ENG–DAN			ENG–SLO		
Method	ML	BT	SRC	HRL	LRL	chrF1	BLEU	rare	chrF1	BLEU	rare	chrF1	BLEU	rare
3-phase scheduled multi-task	✓		✓	✓	✓	51.71	13.94	33.96	50.1	13.7	54.6	50.1	14.1	69.7
2-phase scheduled multi-task	✓		✓	✓	✓	51.42	13.75	33.83	49.8	13.5	55.3	50.2	14.0	69.7
Multi-task w/o schedule	✓		✓	✓	✓	48.62	11.98	29.16	48.0	12.2	52.5	48.3	12.6	68.8
HRL pretraining, LRL fine-tuning	✓		✓	✓	✓	48.15	11.35	29.88	47.8	11.6	49.9	47.0	11.4	66.3
Subnetwork pretraining	✓		✓	✓	✓	47.74	11.17	26.93						

Torrey and Shavlik (2009) describe three ways in which transfer learning can benefit training: 1) higher performance at the very beginning of learning, 2) steeper learning curve, and 3) higher asymptotic performance. When pretraining the encoder and decoder on source and target autoencoder tasks respectively, we see the first of these, but not the other two: for ENG–EST NMT training at first improves faster than with random

initialization, but converges to a worse final model. As the approach was clearly inferior, we did not use it for the other language pairs. However, we have not tested pretraining on a next token prediction or masked language modeling task.

5.4.4 Dataset augmentation – Subword regularization

Table 7 shows an improvement between +0.08 and +0.55 BLEU from using subword regularization as the only noise model, without the use of an autoencoder.

Table 7: Results with subword regularization (SWR).

			Autoencoder			ENG–EST			ENG–DAN			ENG–SLO		
Method	ML	BT	SRC	HRL	LRL	chrF1	BLEU	rare	chrF1	BLEU	rare	chrF1	BLEU	rare
SWR	✓					50.09	12.90	33.20	49.57	13.13	54.21	49.83	13.97	68.79
no SWR	✓					49.77	12.57	31.14	49.27	13.05	53.66	49.27	13.42	69.07

5.4.5 Dataset augmentation – Autoencoder

Table 8 shows an ablation experiment for the noise model. When compared against only using the subword regularization, the additional noises give between +0.2 and +0.5 BLEU. All parts of the noise model are individually ablated: the most important is local reordering, which when omitted causes a decrease of -0.36 BLEU. The full noise model includes subword regularization. When subword regularization is ablated, we turn it entirely off, both for the parallel data and the autoencoder. Word boundary noise, taboo sampling, and insertions are not included in our full noise model, as they did not show a benefit on the development set. However, word boundary noise gives +0.2 BLEU and taboo sampling +0.09 BLEU on the test set.

Table 8: Ablation results for noise model. Ordered by decreasing BLEU.

			Autoencoder			ENG–EST		
Method	ML	BT	SRC	HRL	LRL	chrF-1.0	BLEU	rare
+ Word boundary noise	✓		✓	✓	✓	51.56	13.95	33.20
+ Taboo sampling	✓		✓	✓	✓	51.23	13.84	33.81
No drop	✓		✓	✓	✓	51.48	13.79	33.89
Full noise	✓		✓	✓	✓	51.42	13.75	33.83
+ Insertion	✓		✓	✓	✓	50.88	13.74	33.51
Only switchout	✓		✓	✓	✓	50.78	13.49	32.21
No SWR	✓		✓	✓	✓	50.71	13.46	32.18
Only SWR	✓		✓	✓	✓	50.96	13.43	32.85
No reorder	✓		✓	✓	✓	50.90	13.39	33.03

We also consider for which languages an autoencoder task should be added. Table 9 shows variants starting from no autoencoder, adding autoencoders one by one first for the low-resource target language, then for the source language and finally for the high-resource target language. The best combination uses source and LRL, with the SRC autoencoder giving a gain of +0.11 BLEU over only using the LRL. The HRL autoencoder is detrimental, and leaving it out gives +0.29 BLEU.

5.4.6 Dataset augmentation – Back-translation

Table 10 shows the improvements gained using back-translated synthetic data. We weight the natural and synthetic LRL data equally. Back-translation is generally effective, giving a benefit between +1.31 and +4.46 BLEU. When using back-translated data, the autoencoder task is less effective, with small improvements to Character F_1 but inconsistent results for the other measures. Note that back-translation is not a silver bullet. The *Vanilla BT* system uses only back-translation, but not multilingual training or autoencoder: the back-translation

Table 9: Autoencoder language tasks.

			Autoencoder			ENG–EST		
Method	ML	BT	SRC	HRL	LRL	chrF-1.0	BLEU	rare
SRC+LRL AE	✓		✓		✓	51.71	14.04	34.79
LRL AE	✓				✓	51.41	13.93	33.57
SRC+HRL+LRL AE	✓		✓	✓	✓	51.42	13.75	33.83
No AE	✓					50.09	12.90	33.20

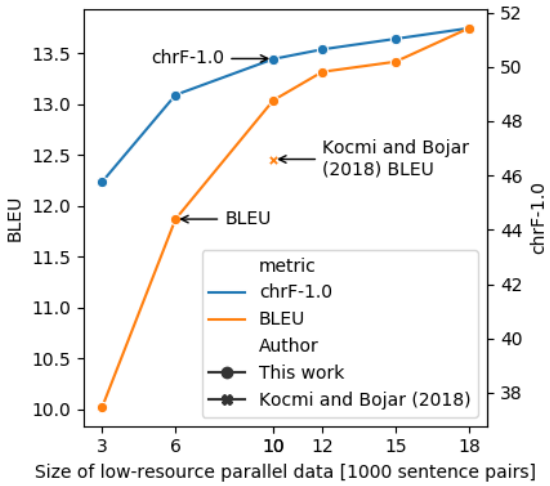


Fig. 7: Varying the amount of low-resource data. Multilingual models, with SRC+HRL+LRL autoencoder and full noise model. Results on English→Estonian newstest2018.

is performed with a weak model trained only on the low-resource parallel data, and then a forward model is trained augmented only by this low-quality back-translation. The performance when using only back-translation is very low: only +2.87 BLEU better than the vanilla model without back-translation. The high-quality back-translation together with multilingual training gives an +12.7 BLEU increase over the vanilla back-translation.

Table 10: Results using back-translation. ✓† indicates the use of a low-quality back-translation made with a non-multilingual non-autoencoder vanilla BT model.

			Autoencoder			ENG–EST			ENG–DAN			ENG–SLO		
Method	ML	BT	SRC	HRL	LRL	chrF1	BLEU	rare	chrF1	BLEU	rare	chrF1	BLEU	rare
Full BT	✓	✓	✓		✓	56.45	18.05	41.13	51.27	14.80	56.63	52.80	16.87	70.97
No AE, full BT	✓	✓				56.33	18.15	40.85	51.20	15.00	57.39	52.65	16.63	70.82
AE, no BT	✓		✓		✓	51.71	14.04	34.79	50.06	13.92	54.58	50.19	14.02	69.94
Vanilla BT		✓†				36.12	5.51	13.25						

5.4.7 Amount of low-resource language data

Figure 7 shows how the performance degrades when the low-resource parallel data is reduced. Each set is subsampled from the previous larger set. All models use multilingual training with scheduled multi-task learning, and SRC+HRL+LRL autoencoders. Down to 10k parallel sentences the performance stays reasonable, after which it rapidly deteriorates.

Also plotted is a 10k sentence pair baseline by Kocmi and Bojar (2018), reaching 12.46 BLEU in a similar setting on the same test set. Our result at 10k is 13.04 BLEU, or +0.68.

5.4.8 Relatedness of the target languages

Table 11 shows the results of using an unrelated but larger HRL (Czech). The results favor transfer from the related HRL (Finnish), by +0.92 BLEU. The difference in favor of the related HRL is largest for the rare words.

Table 11: HRL language relatedness.

			Autoencoder			ENG–EST		
Method	ML	BT	SRC	HRL	LRL	chrF-1.0	BLEU	rare
Within family	FIN		✓		✓	51.71	14.04	34.79
Cross family	CZE		✓		✓	50.20	13.12	30.69

Previously, Zoph et al. (2016) and Dabre et al. (2017) find that related parent languages result in better transfer. However, Kocmi and Bojar (2018) find in the case of Estonian that a bigger parent (Czech) gave better results than a more related parent (Finnish). Our results contradict Kocmi and Bojar (2018) and agree with the prior literature.

5.4.9 Norwegian bokmål → Finnish + North Sámi

We apply the findings of the previous experiments to the low-resource pair Norwegian bokmål to North Sámi. We use a larger task mix weight for the LRL task (40 SRC-HRL / 30 SRC-LRL / 30 BT) to account for the larger LRL parallel data. Table 12 shows the results to be similar to the results of the other languages, with benefit from multilingual training, autoencoder task and back-translation.

Table 12: Results on Norwegian Bokmål–North Sámi Apertium story.

			Autoencoder			NOB–SME		
Method	ML	BT	SRC	HRL	LRL	chrF-1.0	BLEU	rare
ML, AE, BT	✓	✓	✓		✓	57.27	24.40	35.62
ML, AE	✓		✓		✓	54.86	21.07	21.54
Vanilla						45.97	15.64	21.05

5.5 Discussion

In our experiments for four asymmetric-resource one-to-many translation tasks, we find that the largest gains come from cross-lingual transfer (up to +12.7 BLEU), back-translation (up to +4.46 BLEU), and scheduled multi-task learning (up to +2.4 BLEU). To sum up our findings related to the questions asked in the introduction:

On cross-lingual transfer, we find that applying scheduled multi-task learning is superior to both fully sequential and fully parallel transfer. In scheduled multi-task learning, the model is first pretrained on a mix of only high-resource tasks and then fine-tuned using a mix of both high- and low-resource tasks. A second fine-tuning phase only on the low-resource tasks is prone to overfitting.

On exploiting monolingual data, a low-resource target-language autoencoder is beneficial, even when using multilingual training, but inconclusive together with back-translation. A source-language autoencoder is also helpful, to a lesser degree, but a high-resource target autoencoder is not. A noise model including subword regularization, reordering, and deletion is beneficial. The results for substitutions and the proposed taboo sampling method are inconclusive.

On vocabulary construction, Morfessor EM+Prune is superior to SentencePiece in this translation setting, for a gain of +0.6 BLEU. As the methods use the same training algorithm, it indicates that the prior used in Morfessor is beneficial in finding efficient subword lexicons. The vocabulary size has less effect (up to 0.5 BLEU for sizes between 8k and 20k) on the results. Subword lexicon size has been considered an important parameter to tune (Sennrich and Zhang, 2019; Salesky et al., 2020). Also our preliminary experiments of low-resource NMT

without subword regularization suggested a more substantial effect for the lexicon size. It seems that the subword sampling procedure (and perhaps the autoencoder task) lessens the impact of the subword vocabulary size.

Regarding available data and languages, larger low-resource parallel data give better results, but diminishing returns are already reached after 10k sentences. We find language relatedness to be more important than parent language size in highly asymmetrical transfer. Sennrich and Zhang (2019) find that smaller models and batch sizes work better in low-resource settings. We find that large models are better whenever auxiliary multilingual or monolingual data is used. While in the vanilla setting, the smaller model is better, it still falls far behind the models using additional data.

Among the translation tasks, we get the lowest scores in the English–Danish translation. While Danish has the smallest LRL monolingual corpus, as the same order is observed also for the models not using monolingual data, the reason must lie elsewhere, possibly in the difficulty of the JRC-Acquis corpus. The autoencoder task has the largest benefit for English–Estonian. In the Norwegian–North Sámi experiment the size of the low-resource parallel data is an order of magnitude larger than in the other experiments, but the results remain similar. Due to the small size of the test set, we include the entire translation output in Ancillary File `translated.apertium.story.txt`.

The three evaluation measures—BLEU, Character F_1, and rare words F_1—generally agree. Some exceptions include ablation of the subword regularization and using SwitchOut as the sole noise model, which hurt in particular the rare words more than BLEU. Turning off the autoencoder has the least effect on rare words, even giving a slight improvement for ENG–DAN when using back-translation.

Our results again underscore the need to gather parallel data for low-resource language pairs. This may be possible to accomplish at reasonable cost, as 10k sentence pairs already goes a long way. Monolingual corpora of high quality and quantity are also of great importance as auxiliary data for MT.

6 Conclusion

When training a neural translation model for low-resource languages with limited parallel training data, it is important to make use of efficient methods for cross-lingual learning, data augmentation, and subword segmentation. Our experiments in asymmetric-resourced one-to-many translation show that the largest individual improvements come from any cross-lingual transfer learning and augmenting the training data with back-translation. However, considerable benefits are gained also by less common approaches: scheduled multi-task learning, subword regularization, and a denoising autoencoder with multiple noise models. For this reason, we strongly recommend that NMT frameworks should include a dataloader with the ability to (a) sample noisy minibatches for training and (b) use a schedule for controlling the mixing of different tasks. Subword sampling requires a probabilistic segmentation model such as SentencePiece or Morfessor, making them preferable to the more common BPE method. Both our data loader implementation for the OpenNMT-py system and the Morfessor EM+Prune software are available with non-restrictive licenses.

Acknowledgements This study has been supported by the MeMAD project, funded by the European Union's Horizon 2020 research and innovation programme (grant agreement No 780069), and the FoTran project, funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 771113). Computer resources within the Aalto University School of Science "Science-IT" project were used.

References

Arivazhagan N, Bapna A, Firat O, Lepikhin D, Johnson M, Krikun M, Chen MX, Cao Y, Foster G, Cherry C, Macherey W, Chen Z, Wu Y (2019) Massively multilingual neural machine translation in the wild: Findings and challenges. URL `http://arxiv.org/abs/1907.05019`, arXiv:1907.05019 [cs.CL], `1907.05019`

Artetxe M, Labaka G, Agirre E, Cho K (2018) Unsupervised neural machine translation. In: Proceedings of the 6th International Conference on Learning Representations (ICLR), URL `http://arxiv.org/abs/1710.11041`

Belinkov Y, Bisk Y (2017) Synthetic and natural noise both break neural machine translation. URL `https://arxiv.org/abs/1711.02173`, arXiv:1711.02173 [cs.CL]

Blackwood G, Ballesteros M, Ward T (2018) Multilingual neural machine translation with task-specific attention. In: Proceedings of the 27th International Conference on Computational Linguistics, pp 3112–3122

Bojar O, Dušek O, Kocmi T, Libovický J, Novák M, Popel M, Sudarikov R, Variš D (2016) CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In: Sojka P, Horák A, Kopeček I, Pala K

(eds) Text, Speech, and Dialogue: 19th International Conference, TSD 2016, Masaryk University, Springer International Publishing, Cham / Heidelberg / New York / Dordrecht / London, no. 9924 in Lecture Notes in Artificial Intelligence, pp 231–238

Bojar O, Federmann C, Fishel M, Graham Y, Haddow B, Huck M, Koehn P, Monz C (2018) Findings of the 2018 conference on machine translation (wmt18). In: Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers, Association for Computational Linguistics, Belgium, Brussels, pp 272–307, URL http://www.aclweb.org/anthology/W18-6401

Bourlard H, Kamp Y (1988) Auto-association by multilayer perceptrons and singular value decomposition. Biological cybernetics 59(4-5):291–294

Caruana R (1998) Multitask learning. In: Learning to learn, Springer, pp 95–133

Caswell I, Chelba C, Grangier D (2019) Tagged back-translation. In: Proceedings of the Fourth Conference on Machine Translation (WMT) (Volume 1: Research Papers), pp 53–63

Chen Z, Badrinarayanan V, Lee CY, Rabinovich A (2018) Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In: Proceedings of the International Conference on Machine Learning (ICML), pp 794–803

Cheng Y, Xu W, He Z, He W, Wu H, Sun M, Liu Y (2016) Semi-supervised learning for neural machine translation. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers), pp 1965–1974, URL http://arxiv.org/abs/1606.04596

Cherry C, Foster G, Bapna A, Firat O, Macherey W (2018) Revisiting character-based neural machine translation with capacity and compression. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Brussels, Belgium, pp 4295–4305, DOI 10.18653/v1/D18-1461, URL https://www.aclweb.org/anthology/D18-1461

Chu C, Dabre R, Kurohashi S (2017) An empirical comparison of domain adaptation methods for neural machine translation. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 2: Short Papers), Association for Computational Linguistics, Vancouver, Canada, pp 385–391, DOI 10.18653/v1/P17-2061, URL https://www.aclweb.org/anthology/P17-2061

Chung J, Cho K, Bengio Y (2016) A character-level decoder without explicit segmentation for neural machine translation. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers), pp 1693–1703, URL http://arxiv.org/abs/1603.06147

Conneau A, Lample G (2019) Cross-lingual language model pretraining. In: Proceedings of Advances in neural information processing systems (NIPS), pp 7059–7069, URL http://papers.nips.cc/paper/8928-cross-lingual-language-model-pretraining

Costa-jussà MR, Fonollosa JAR (2016) Character-based neural machine translation. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 2: Short Papers), Association for Computational Linguistics, Berlin, Germany, pp 357–361, DOI 10.18653/v1/P16-2058, URL https://www.aclweb.org/anthology/P16-2058

Costa-jussà MR, Escolano C, Fonollosa JAR (2017) Byte-based neural machine translation. In: Proceedings of the First Workshop on Subword and Character Level Models in NLP, Association for Computational Linguistics, Copenhagen, Denmark, pp 154–158, DOI 10.18653/v1/W17-4123, URL https://www.aclweb.org/anthology/W17-4123

Creutz M, Lagus K (2002) Unsupervised discovery of morphemes. In: Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning (MPL), Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, vol 6, pp 21–30, DOI 10.3115/1118647.1118650, URL http://portal.acm.org/citation.cfm?doid=1118647.1118650

Creutz M, Lagus K (2005) Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. Technical Report A81, Publications in Computer and Information Science, Publications in Computer and Information Science, Helsinki University of Technology

Creutz M, Lagus K (2007) Unsupervised models for morpheme segmentation and morphology learning. ACM Transactions on Speech and Language Processing 4(1)

Currey A, Miceli-Barone AV, Heafield K (2017) Copied monolingual data improves low-resource neural machine translation. In: Proceedings of the Second Conference on Machine Translation (WMT), pp 148–156

Dabre R, Nakagawa T, Kazawa H (2017) An empirical study of language relatedness for transfer learning in neural machine translation. In: Proceedings of the 31st Pacific Asia Conference on Language, Information and Computation, pp 282–286

Dabre R, Chu C, Kunchukuttan A (2020) A comprehensive survey of multilingual neural machine translation. ArXiv:2001.01115 [cs.CL], 2001.01115

Dai AM, Le QV (2015) Semi-supervised sequence learning. In: Proceedings of Advances in neural information processing systems (NIPS), pp 3079–3087

Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B (Methodological) 39(1):1–38

Devlin J, Chang MW, Lee K, Toutanova K (2019) BERT: Pre-training of deep bidirectional Transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp 4171–4186, URL http://arxiv.org/abs/1810.04805

Di Gangi MA, Federico M (2017) Monolingual embeddings for low resourced neural machine translation. In: Proceedings of the 14th International Workshop on Spoken Language Translation (IWSLT'17), pp 97–104

Domhan T, Hieber F (2017) Using target-side monolingual data for neural machine translation through multi-task learning. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 1500–1505

Edunov S, Ott M, Auli M, Grangier D (2018) Understanding back-translation at scale. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 489–500

Firat O, Cho K, Bengio Y (2016) Multi-way, multilingual neural machine translation with a shared attention mechanism. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp 866–875, URL http://arxiv.org/abs/1601.01073

Forcada ML, Ginestí-Rosell M, Nordfalk J, O'Regan J, Ortiz-Rojas S, Pérez-Ortiz JA, Sánchez-Martínez F, Ramírez-Sánchez G, Tyers FM (2011) Apertium: a free/open-source platform for rule-based machine translation. Machine translation 25(2):127–144

Gage P (1994) A new algorithm for data compression. The C Users Journal 12(2):23–38

Galuščáková P, Bojar O (2012) WMT 2011 testing set. URL http://hdl.handle.net/11858/00-097C-0000-0006-AADA-9, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University

Goldsmith J (2001) Unsupervised learning of the morphology of a natural language. Computational Linguistics 27(2):153–198, DOI 10.1162/089120101750300490, URL http://www.mitpressjournals.org/doi/10.1162/089120101750300490

Goodfellow IJ, Mirza M, Xiao D, Courville A, Bengio Y (2014) An empirical investigation of catastrophic forgetting in gradient-based neural networks. In: Proceedings of International Conference on Learning Representations (ICLR), Citeseer, URL https://arxiv.org/abs/1312.6211

Graça M, Kim Y, Schamper J, Khadivi S, Ney H (2019) Generalizing back-translation in neural machine translation. In: Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers), pp 45–52, URL https://arxiv.org/abs/1906.07286

Grönroos SA, Virpioja S, Kurimo M (2018) Cognate-aware morphological segmentation for multilingual neural translation. In: Proceedings of the Third Conference on Machine Translation, Association for Computational Linguistics, Brussels, Belgium

Grönroos SA, Virpioja S, Kurimo M (2020) Morfessor EM+Prune: Improved subword segmentation with expectation maximization and pruning. In: Proceedings of the 12th Language Resources and Evaluation Conference, ELRA, Marseilles, France

Gu J, Wang Y, Chen Y, Cho K, Li VOK (2018) Meta-learning for low-resource neural machine translation. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 3622–3631, URL http://arxiv.org/abs/1808.08437

Gulcehre C, Firat O, Xu K, Cho K, Barrault L, Lin HC, Bougares F, Schwenk H, Bengio Y (2015) On using monolingual corpora in neural machine translation. URL http://arxiv.org/abs/1503.03535

Hammarström H, Borin L (2011) Unsupervised learning of morphology. Computational Linguistics 37(2):309–350

Harris ZS (1955) From phoneme to morpheme. Language 31(2):190–222

He D, Xia Y, Qin T, Wang L, Yu N, Liu TY, Ma WY (2016) Dual learning for machine translation. In: Proceedings of Advances in neural information processing systems (NIPS), pp 820–828, URL http://arxiv.org/abs/1611.00179

Iyyer M, Manjunatha V, Boyd-Graber J, Daumé III H (2015) Deep unordered composition rivals syntactic methods for text classification. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP) (Volume 1: Long Papers), pp 1681–1691

Johnson M, Schuster M, Le QV, Krikun M, Wu Y, Chen Z, Thorat N, Viégas F, Wattenberg M, Corrado

G, Hughes M, Dean J (2017) Google's multilingual neural machine translation system: Enabling zero-shot translation. Transactions of the Association for Computational Linguistics 5:339–351, URL http://arxiv.org/abs/1611.04558

Joshi M, Chen D, Liu Y, Weld DS, Zettlemoyer L, Levy O (2020) Spanbert: Improving pre-training by representing and predicting spans. Transactions of the Association for Computational Linguistics 8:64–77

Kalchbrenner N, Blunsom P (2013) Recurrent continuous translation models. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 1700–1709

Karakanta A, Dehdari J, van Genabith J (2018) Neural machine translation for low-resource languages without parallel corpora. Machine Translation 32(1-2):167–189

Kiperwasser E, Ballesteros M (2018) Scheduled multi-task learning: From syntax to translation. Transactions of the Association for Computational Linguistics 6:225–240

Klein G, Kim Y, Deng Y, Senellart J, Rush AM (2017) OpenNMT: Open-source toolkit for neural machine translation. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), DOI 10.18653/v1/P17-4012, URL http://arxiv.org/abs/1701.02810, arXiv: 1701.02810

Kocmi T (2019) Exploring benefits of transfer learning in neural machine translation. PhD Thesis, Charles University

Kocmi T, Bojar O (2018) Trivial transfer learning for low-resource neural machine translation. In: Proceedings of the Third Conference on Machine Translation (WMT): Research Papers, pp 244–252

Koehn P (2005) Europarl: A parallel corpus for statistical machine translation. In: MT summit, vol 5, pp 79–86

Kohonen O, Virpioja S, Lagus K (2010) Semi-supervised learning of concatenative morphology. In: Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology, Association for Computational Linguistics, Uppsala, Sweden, pp 78–86, URL http://www.aclweb.org/anthology/W10-2210

Koponen M, Salmi L, Nikulin M (2019) A product and process analysis of post-editor corrections on neural, statistical and rule-based machine translation output. Machine Translation 33(1-2):61–90

Kreutzer J, Sokolov A (2018) Learning to segment inputs for NMT favors character-level processing. In: Proceedings of the 15th International Workshop on Spoken Language Translation (IWSLT), URL https://arxiv.org/abs/1810.01480

Kudo T (2018) Subword regularization: Improving neural network translation models with multiple subword candidates. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers), pp 66–75, URL http://arxiv.org/abs/1804.10959

Kudo T, Richardson J (2018) SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Brussels, Belgium, pp 66–71, DOI 10.18653/v1/D18-2012, URL https://www.aclweb.org/anthology/D18-2012

Kurimo M, Virpioja S, Turunen V, Lagus K (2010) Morpho challenge 2005-2010: Evaluations and results. In: Heinz J, Cahill L, Wicentowski R (eds) Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology, Association for Computational Linguistics, Uppsala, Sweden, pp 87–95

Lample G, Conneau A, Denoyer L, Ranzato M (2018a) Unsupervised machine translation using monolingual corpora only. In: International Conference on Learning Representations (ICLR), URL http://arxiv.org/abs/1711.00043

Lample G, Ott M, Conneau A, Denoyer L, Ranzato M (2018b) Phrase-based & neural unsupervised machine translation. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 5039–5049, URL https://www.aclweb.org/anthology/D18-1549.pdf

Lee YS (2004) Morphological analysis for statistical machine translation. In: Proceedings of HLT-NAACL 2004: Short Papers, Association for Computational Linguistics, pp 57–60

Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, Stoyanov V, Zettlemoyer L (2019) BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. ArXiv:1910.13461 [cs.CL], 1910.13461

Lison P, Tiedemann J (2016) OpenSubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In: Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016), European Language Resources Association

Luong MT (2016) Neural machine translation. PhD Thesis, Stanford University

Luong MT, Le QV, Sutskever I, Vinyals O, Kaiser L (2015) Multi-task sequence to sequence learning. In: Proceedings of International Conference on Learning Representations (ICLR), URL http://arxiv.org/abs/

1511.06114
McCloskey M, Cohen NJ (1989) Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of learning and motivation, vol 24, Elsevier, pp 109–165
Mueller A, Nicolai G, McCarthy AD, Lewis D, Wu W, Yarowsky D (2020) An analysis of massively multilingual neural machine translation for low-resource languages. In: Proceedings of The 12th Language Resources and Evaluation Conference, pp 3710–3718
Oflazer K, El-Kahlout ID (2007) Exploring different representational units in English-to-Turkish statistical machine translation. In: Proceedings of the second workshop on statistical machine translation, Association for Computational Linguistics, pp 25–32, DOI 10.3115/1626355.1626359, URL http://portal.acm.org/citation.cfm?doid=1626355.1626359
Papineni K, Roukos S, Ward T, Zhu WJ (2002) BLEU: a method for automatic evaluation of machine translation. In: 40th annual meeting of the association for computational linguistics, Association for Computational Linguistics, Philadelphia, PA, USA, pp 311–318, DOI 10.3115/1073083.1073135, URL http://portal.acm.org/citation.cfm?doid=1073083.1073135
Platanios EA, Sachan M, Neubig G, Mitchell T (2018) Contextual parameter generation for universal neural machine translation. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 425–435, URL https://www.aclweb.org/anthology/D18-1039
Popović M (2015) chrF: character n-gram F-score for automatic MT evaluation. In: Proceedings of the 10th Workshop on Statistical Machine Translation (WMT), Association for Computational Linguistics, pp 392–395, DOI 10.18653/v1/W15-3049, URL http://aclweb.org/anthology/W15-3049
Ramachandran P, Liu PJ, Le Q (2017) Unsupervised pretraining for sequence to sequence learning. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 383–391
Rissanen J (1989) Stochastic Complexity in Statistical Inquiry, vol 15. World Scientific Series in Computer Science, Singapore
Sachan DS, Neubig G (2018) Parameter sharing methods for multilingual self-attentional translation models. In: Proceedings of the Third Conference on Machine Translation (WMT): Research Papers, pp 261–271, URL https://www.aclweb.org/anthology/W18-6327
Salesky E, Runge A, Coda A, Niehues J, Neubig G (2020) Optimizing segmentation granularity for neural machine translation. Machine Translation pp 1–19
Scott SL (2002) Bayesian methods for hidden Markov models: Recursive computing in the 21st century. Journal of the American Statistical Association 97(457):337–351
Sennrich R, Zhang B (2019) Revisiting low-resource neural machine translation: A case study. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp 211–221
Sennrich R, Haddow B, Birch A (2015) Neural machine translation of rare words with subword units. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), URL http://arxiv.org/abs/1508.07909
Sennrich R, Haddow B, Birch A (2016) Improving neural machine translation models with monolingual data. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp 86–96
Skorokhodov I, Rykachevskiy A, Emelyanenko D, Slotin S, Ponkratov A (2018) Semi-supervised neural machine translation with language models. In: Proceedings of the AMTA 2018 Workshop on Technologies for MT of Low Resource Languages (LoResMT 2018), pp 37–44
Song K, Tan X, Qin T, Lu J, Liu TY (2019) Mass: Masked sequence to sequence pre-training for language generation. In: Proceedings of the International Conference on Machine Learning (ICML), pp 5926–5936
Sriram A, Jun H, Satheesh S, Coates A (2017) Cold fusion: Training seq2seq models together with language models. In: Proceedings of the Interspeech 2018, URL https://arxiv.org/abs/1708.06426
Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1):1929–1958
Stahlberg F, Cross J, Stoyanov V (2018) Simple fusion: Return of the language model. In: Proceedings of the Third Conference on Machine Translation (WMT): Research Papers, pp 204–211
Steinberger R, Pouliquen B, Widiger A, Ignat C, Erjavec T, Tufiş D, Varga D (2006) The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In: Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006), Genoa, Italy
Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: Proceedings of Advances in neural information processing systems (NIPS), pp 3104–3112
Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer

vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2818–2826, URL http://arxiv.org/abs/1512.00567

Thompson B, Khayrallah H, Anastasopoulos A, McCarthy AD, Duh K, Marvin R, McNamee P, Gwinnup J, Anderson T, Koehn P (2018) Freezing subnetworks to analyze domain adaptation in neural machine translation. In: Proceedings of the Third Conference on Machine Translation (WMT): Research Papers, pp 124–132

Tiedemann J (2009) Character-based PSMT for closely related languages. In: Proceedings of the 13th Conference of the European Association for Machine Translation (EAMT 2009), pp 12–19

Toral A, Sánchez-Cartagena VM (2017) A multifaceted evaluation of neural versus phrase-based machine translation for 9 language directions. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL): Volume 1, Long Papers, pp 1063–1073, URL https://www.aclweb.org/anthology/E17-1100

Torrey L, Shavlik J (2009) Transfer learning. In: Olivas ES (ed) Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques: Algorithms, Methods, and Techniques, IGI Global, pp 242–264

Tu Z, Liu Y, Shang L, Liu X, Li H (2017) Neural machine translation with reconstruction. In: Thirty-First AAAI Conference on Artificial Intelligence, URL http://arxiv.org/abs/1611.01874

Vaibhav V, Singh S, Stewart C, Neubig G (2019) Improving robustness of machine translation with synthetic noise. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp 1916–1920, URL https://www.aclweb.org/anthology/N19-1190/

Varjokallio M, Kurimo M, Virpioja S (2013) Learning a subword vocabulary based on unigram likelihood. In: Proceedings of the 2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), IEEE, Olomouc, Czech Republic, pp 7–12, DOI 10.1109/ASRU.2013.6707697, URL http://ieeexplore.ieee.org/document/6707697/

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. In: Advances in Neural Information Processing Systems, pp 6000–6010, URL http://arxiv.org/abs/1706.03762

Vincent P, Larochelle H, Bengio Y, Manzagol PA (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning (ICML), pp 1096–1103

Virpioja S, Väyrynen JJ, Creutz M, Sadeniemi M (2007) Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. In: Machine Translation Summit XI, Copenhagen, Denmark, vol 2007, pp 491–498

Virpioja S, Turunen VT, Spiegler S, Kohonen O, Kurimo M (2011) Empirical comparison of evaluation methods for unsupervised learning of morphology. Traitement Automatique des Langues 52(2):45–90, URL http://www.atala.org/Empirical-Comparison-of-Evaluation

Virpioja S, Smit P, Grönroos SA, Kurimo M (2013) Morfessor 2.0: Python implementation and extensions for Morfessor Baseline. Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, Department of Signal Processing and Acoustics, Aalto University, Helsinki, Finland

Wang X, Pham H, Dai Z, Neubig G (2018) Switchout: an efficient data augmentation algorithm for neural machine translation. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 856–861, URL https://www.aclweb.org/anthology/D18-1100

Yang Z, Chen W, Wang F, Xu B (2018) Unsupervised neural machine translation with weight sharing. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp 46–55

Zhang J, Zong C (2016) Exploiting source-side monolingual data in neural machine translation. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 1535–1545

Zoph B, Yuret D, May J, Knight K (2016) Transfer learning for low-resource neural machine translation. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 1568–1575

Östling R, Tiedemann J (2017) Neural machine translation for low-resource languages. ArXiv:1708.05729 [cs.CL], 1708.05729

Publication X

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. Extending hybrid word-character neural machine translation with multi-task learning of morphological analysis. In *Proceedings of the Second Conference on Machine Translation*, Copenhagen, Denmark, pages 296–302, Sep 2017.

Extending hybrid word-character neural machine translation with multi-task learning of morphological analysis

Stig-Arne Grönroos and **Sami Virpioja** and **Mikko Kurimo**
stig-arne.gronroos@aalto.fi
Department of Signal Processing and Acoustics, Aalto University, Finland

Abstract

This article describes the Aalto University entry to the English-to-Finnish news translation shared task in WMT 2017. Our system is an open vocabulary neural machine translation (NMT) system, adapted to the needs of a morphologically complex target language. The main contributions of this paper are 1) implicitly incorporating morphological information to NMT through multi-task learning, 2) adding an attention mechanism to the character-level decoder, combined with character segmentation of names, and 3) a new overattending penalty to beam search.

1 Introduction

The rich inflection, derivation and compounding in synthetic languages can result in very large vocabularies. In statistical machine translation (SMT) large vocabularies cause sparsity issues. While continuous space representations make neural machine translation (NMT) more robust towards such sparsity, it suffers from a different set of problems related to large vocabularies. A large vocabulary bloats memory and computation requirements, while still leaving the problem of out-of-vocabulary words unsolved.

Subword vocabularies have been proposed as a solution. While the benefits of using subwords in SMT have been at best moderate (Virpioja et al., 2007; Fishel and Kirik, 2010; Grönroos et al., 2015), subword decoding has become popular in NMT (Sennrich et al., 2015). A subword vocabulary of a moderate size ensures full coverage of an open vocabulary. The downside is an increase in the length of the input and output sequences. Long sequences cause a large increase in computation time, especially for architectures using the attention mechanism.

An alternative approach is the hybrid word-character decoder presented by Luong and Manning (2016). In the hybrid decoder, a word level decoder outputs frequent words as they are, while replacing infrequent words with a special <UNK> symbol. A second character-level decoder then expands these <UNK> symbols into surface forms.

In addition to providing moderate length of input and output sequences together with an open vocabulary, the hybrid word-character decoder makes it simple to use labels based on the level of words, provided for example by morphological analyzers and parsers. In SMT, such tools are typically used via factored translation models (Koehn and Hoang, 2007). Factored translation has also been successfully applied in NMT. For example, Sennrich and Haddow (2016) augment the source words with four additional factors: PoS, lemma, dependency label and subwords. García-Martínez et al. (2016) use a decomposed generation process, in which they first output lemma, PoS, tense, person, gender, and number, from which the surface form is generated using a rule-based morphological analyzer.

Neural machine translation provides another way to utilize external annotations, multi-task learning (MTL). MTL is a well established machine learning approach that aims at improving the generalization performance of a task using other related tasks (Caruana, 1998). For example, Luong et al. (2016) use autoencoding, parsing, and caption generation as auxiliary tasks to improve English-to-German translation. Eriguchi et al. (2017) combine NMT with a Recurrent Neural Network Grammar. The system learns to parse the target language as an auxiliary task when translating into English.

We propose an MTL approach inspired by fac-

Proceedings of the Conference on Machine Translation (WMT), Volume 2: Shared Task Papers, pages 296–302
Copenhagen, Denmark, September 711, 2017.

tored translation. The output of a morphological analyzer for the target sentence is used as an auxiliary prediction target, while sharing network parameters to a larger extent than in the approach of Luong et al. (2016).

This approach has two advantages over factored models. When training a system using factored output, embedded gold standard labels are given as input to the decoder. During translation gold standard labels are not available, and predicted labels are instead fed back in. The confidence of the predictions is not accounted for when feeding back the labels. This might worsen the problems caused by exposure bias, i.e., the mismatch between training and inference (Ranzato et al., 2016). If factored input is used, the external labeling tools need to be included also in the translation pipeline. In MTL such tools are only necessary during training.

In terms of computational cost, a factored model needs to predict the auxiliary labels also during translation, slowing down inference and complicating the beam search. A factored model might also need to use a larger beam to avoid hypotheses with the same surface form but different labels from crowding out more diverse hypotheses. In MTL, the auxiliary tasks are only performed during training, and no changes need to be made to the inference.

The main contributions of this paper are combining word-level labels from morphological analysis with a hybrid word-character decoder, and adding an attention mechanism to the character-level decoder. We also propose a new overattending penalty to the beam search.

2 Neural machine translation

Neural machine translation (NMT) is a framework for machine translation that uses a single neural network trained end-to-end. The recently proposed encoder-decoder network with attention mechanism (Bahdanau et al., 2014) has become accepted as the current standard in NMT.

The first part of the network, the encoder, reads a source sentence $\boldsymbol{x}$ and encodes it as a sequence of hidden states $s = (s_1, s_2, \ldots, s_N)$. The encoder is often implemented as a bidirectional recurrent network with long short-term memory units (bi-LSTM), in which case each hidden state is the concatenation of a state from the forward and backward encoders.

The last part of the network, the decoder, is implemented as a conditional recurrent language model which models the probability of the target sentence $\boldsymbol{y}$ as

$$\begin{aligned} \log p(\boldsymbol{y} \mid \boldsymbol{x}) &= \sum_t \log p(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) \\ &= \sum_t \log p(y_t \mid h_t, c_t). \end{aligned} \quad (1)$$

The encoder and decoder are linked by the attention mechanism. At each timestep, the attention mechanism computes a context vector c_t as a weighted average of the encoder hidden states s. The weights $a_{t,i}$ are determined by a layer that takes as input the current decoder hidden state h_t and each of the vectors s_i in turn.

$$\begin{aligned} a_{t,i}(h, s) &= \frac{\exp(\text{align}(\boldsymbol{h}_t, \boldsymbol{s}_i))}{\sum_j \exp(\text{align}(\boldsymbol{h}_t, \boldsymbol{s}_j))} \\ \text{align}(\boldsymbol{h}_t, \boldsymbol{s}_i) &= \boldsymbol{v}_a^\top \tanh(W_a[\boldsymbol{h}_t; \boldsymbol{s}_i]) \end{aligned} \quad (2)$$

In effect, at each timestep the attention mechanism scans the entire source to decide which parts are relevant to focus on when generating the next output symbol.

Luong and Manning (2016) extend the word-level encoder-decoder model by adding character-level processing of rare words. On the encoder side, word embeddings for rare source words are produced by a character-level encoder, instead of using a universal <UNK> embedding. The hybrid model ensures an open vocabulary, while keeping the attended sequence shorter than using characters or subwords.

On the decoder side, the word-level decoder outputs <UNK> for rare words, while storing the decoder hidden state at that timestep. A separate character-level decoder expands these tokens into the surface form. The character-level encoder and decoder can be trained jointly with the word-level components, by backpropagating end-to-end.

In separate-path initialization of the character-level decoder, the word-level LSTM output h is not used to seed the character-level decoder, but instead a counterpart vector $\breve{h}$ is calculated as

$$\breve{h}_t = \tanh(\breve{W}[c_t; h_t])$$

3 System description

Our system is based on the open-source Helsinki Neural Machine Translation (HNMT) software[1].

[1] Available from https://github.com/robertostling/hnmt .

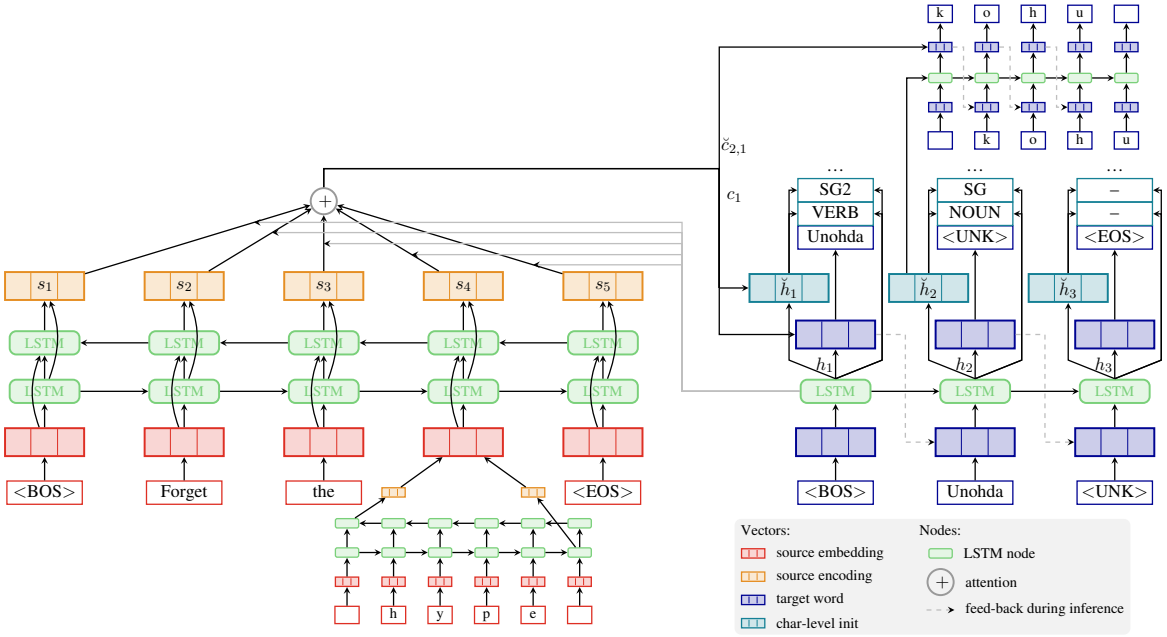


Figure 1: Our neural network architecture. In the example, "Forget the hype" is translated into "Unohda kohu". On the left side, the hybrid word-character encoder, using bi-LSTM for both levels. On the lower right side, the word-level attentional LSTM decoder, which predicts both word tokens and auxiliary labels. Above it, the predicted <UNK> is expanded by the attentional character-level decoder. For clarity, attention is only drawn for the first timestep of each decoder.

We extend[2] HNMT with a hybrid word-character decoder, multi-task learning, and improved beam search. An overview of the neural network architecture can be seen in Figure 1.

Hybrid encoder-decoder. HNMT implements a hybrid word-character encoder. Instead of the two-level unidirectional LSTM character-level encoders of Luong and Manning (2016), bi-LSTM encoders are used. The embedding for rare words is the concatenation of the last states of the forward and backward character-level encoders.

We extend HNMT with a hybrid word-character decoder, using separate path initialization of the character-level decoder. We also add an attention mechanism to the character-level decoder, yielding the character-level context vector $\check{c}_{t,t_c}$. The attended sequence is the same as for the word-level decoder: the word-level encoding s of the source sentence. To make it possible for the attentional character-level decoder to copy or transcribe on a subword-level, we perform character segmentation preprocessing on capitalized input words (after truecasing). The segmentation is described in Section 4.

Multi-task learning. The main task is translation into the target language surface form, while the auxiliary tasks consist of predicting the output of the FinnPos morphological analyzer for the target sentence. The auxiliary tasks provide additional supervision signals that can help the model learn grammar and morphology. The tasks share parameters more closely than the one-to-many multi-task learning setting defined by Luong et al. (2016). In addition to sharing the encoder, all parts of the word level decoder except the final feed-forward prediction layers are shared. A potential downside compared to using a separate decoder is that the label sequence must be of the same length and synchronous with the surface sequence. This tightly shared MTL matches perfectly with the hybrid word-character decoder, as the labeling is on the level of words. The work-around of repeating labels to match the length of a subword sequence was not explored in this work.

In MTL, the supervision from the labels is softer than when using a factored model. Uncertain labels could be ignored, by limiting the task to sentences with high-confidence labels. We did not use this opportunity, as FinnPos labels every input sentence, and does not provide confidence estimates. As all our data $\mathcal{D}$ is labeled, we control the influence of the auxiliary task using a multiplica-

[2] Our fork available from https://github.com/Waino/hnmt .

tive weight on part of the cost function, instead of the minibatch mixing ratio used by Luong et al. (2016).

We train the whole model jointly to maximize

$$E_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{a})\in\mathcal{D}}[\log p(\boldsymbol{y},\boldsymbol{a}\mid\boldsymbol{x})]$$

where $\boldsymbol{a}$ are the labels: the cluster id of the lemma, the rounded log-frequency of the lemma, the PoS, and 5 morphological tags: number, case, person, mood, and tense. Each label is independently predicted from the concatenation of h and $\breve{h}$.

Beam search scoring function. We use beam search during decoding to find the optimal translation sequence $\boldsymbol{y}$. Instead of directly maximizing the probability, we maximize a score function $s(\boldsymbol{y},\boldsymbol{x})$, designed to alleviate two known issues in NMT: overtranslation and undertranslation.

Undertranslation is reduced by adding length normalization (lp) and a coverage penalty (cp), following Wu et al. (2016).

Unlike undertranslation, overtranslation is to some extent inherently reduced by the monotonically increasing generation log-probabilities. However, the inherent cost is not enough, leading us to add a penalty for overattending a source token (oap). The penalty is applied if the most attended source word has sum attention over 1.0. We use the maximum function instead of sum, in order not to increase the strength of the penalty for long input sentences. The overattending penalty is monotonically increasing, which enables us to include it when pruning active hypotheses.

The overattending penalty is not suitable if the decoder uses smaller units than the output of the encoder. Repeated attention is required if the decoder must output several subwords for each source token.

The scoring function is

$$\begin{aligned} s(\boldsymbol{y},\boldsymbol{x}) = &-\log\big(p(\boldsymbol{y}\mid\boldsymbol{x})\big) + \mathrm{lp}(\boldsymbol{y}) \\ &+ \mathrm{cp}(\boldsymbol{y},\boldsymbol{x}) + \mathrm{oap}(\boldsymbol{y},\boldsymbol{x}), \end{aligned} \tag{3}$$

where

$$\mathrm{lp}(\boldsymbol{y}) = \frac{(|\boldsymbol{y}|+\lambda)^\alpha}{(1+\lambda)^\alpha} \tag{4}$$

$$\mathrm{cp}(\boldsymbol{y},\boldsymbol{x}) = \beta\sum_{i=1}^{|\boldsymbol{x}|}\log\big(\min(\sum_{j=1}^{|\boldsymbol{y}|}a_{ij},1.0)\big) \tag{5}$$

$$\mathrm{oap}(\boldsymbol{y},\boldsymbol{x}) = -\gamma\max\Big(\max_{i=1}^{|\boldsymbol{x}|}\big(\sum_{j=1}^{|\boldsymbol{y}|}a_{ij}-1.0\big),0.0\Big) \tag{6}$$

The parameters α,β,γ, and λ control the strengths of the penalties.

Pruning in beam search. We use three types of pruning in the beam search.

First, at each step, for each hypothesis to be extended, we prune the list of candidates for the next symbol based on local probability, to only keep $beam\_width + 1$ candidates. This pruning improves speed without affecting the output.

Second, after at least one hypothesis has been completed, we keep track of the current best normalized score. This allows pruning active hypotheses by comparing their partially normalized score against the best normalized score, with adjustable pruning margin. The partially normalized score is calculated as the sum of the monotonically increasing parts of the scoring function

$$-\log\big(p(\boldsymbol{y}\mid\boldsymbol{x})\big) + \mathrm{oap}(\boldsymbol{y},\boldsymbol{x})$$

This pruning may affect the output by removing a hypothesis with a poor early score that could have improved later. To gain a speed-up, it is necessary to prune active hypotheses: limiting pruning to completed hypotheses cannot reduce the number of hypotheses in early stages, and thus cannot result in early clearing of the beam.

Completed hypotheses are moved from the beam to a separate heap. This clears out room in the beam for active hypotheses, but also means that the pruning of active hypotheses becomes essential for early stopping of the beam search.

The third type of pruning is applied to the heap of completed hypotheses based on normalized score, to only keep n best hypotheses. This pruning conserves memory and does not affect the ordering of the results.

4 Data

Our system participates in the constrained condition of the WMT shared task. As training data, we used the Europarl-v8, Rapid and Wikititles corpora, extended with backtranslated monolingual data, resulting in 6 091 184 parallel sentence pairs after cleaning. The backtranslated sentences were from the news.2014.fi corpus, translated with a PB-SMT model, trained with WMT16 constrained settings. Based on initial experiments we decided to use the full backtranslated set, for a ratio of ca 60% backtranslated to 40% parallel data, instead of subsampling to balance the ratio.

Configuration	newstest2016AB			newstest2017			
	chrF-1	chrF-2	BLEU	chrF-1	chrF-2	BLEU	TER
Hybrid decoder with MTL, ensemble of 4	56.79	**55.60**	21.46	57.30	**55.96**	20.28	**.673**
+ repetition removal	**57.07**	55.59	**21.55**	**57.57**	55.92	**20.31**	–
FlatCat subword decoder, ensemble of 4	55.77	55.41	20.01	54.10	53.98	17.15	.750
Hybrid decoder with MTL, single model	54.69	53.43	18.60	55.17	53.87	17.84	–

Table 1: Results of automatic evaluation. BLEU and chrF scores are percentages. TER from http://matrix.statmt.org/matrix/systems_list/1871?metric_id=2 .

Configuration	newstest2016AB		
	chrF-1	chrF-2	BLEU
Hybrid decoder with MTL	56.79	55.60	21.46
No morphological tags	55.97	55.20	19.83
No log frequency	55.49	54.26	19.47
No clustered lemma	55.23	53.65	19.37
No PoS-tags	55.05	53.73	19.29
No multi-task learning	54.91	53.48	19.43
No character attention & name segmentation	52.12	50.80	17.16
No length penalty	56.68	55.52	21.35
No overattending penalty	56.68	55.53	21.33
No coverage penalty	56.43	54.93	20.97
No penalties	55.90	54.21	20.45

Table 2: Results of ablation experiments. All runs are ensembles of 4, to reduce variability.

Data preprocessing consists of filtering too long sentences, normalizing misencoded data, normalizing punctuation, deduplication, tokenization, statistical truecasing, filtering of untranslated sentences, and character segmentation of names on the source side.

Segmenting names into characters, when combined with attention on the character level, allows copying or transliteration on a character-to-character basis. It is applied using a rough heuristic: we segment any token longer than one character beginning with an upper case letter or digit. All segmented characters are marked using reserved symbols. The first and last characters of the sequence have distinct symbols separating them from word-internal characters.

The filtering of untranslated sentences was also performed using a rough heuristic, by filtering any sentences containing certain common English contractions and clitics that do not occur in Finnish. The target side training data, especially Europarl, contains hundreds of sentences with English phrases. A typical reason is discussions on the wording of English-language documents being drafted. The filtering was an attempt to alleviate a failure mode in which the system would instead of translating attempt (and fail) to output the English source.

A parallel corpus augmented with gold-standard labels for MTL is not available. We tag the target side of the parallel corpus using the statistical tagger FinnPos (Silfverberg et al., 2016). The resulting labels are noisy, but nonetheless provide supervision for the morphological analysis task.

We postprocess the output of FinnPos. The morpheme tag sequence is split, and tags are grouped by type. FinnPos lemmas are noisy, containing many remaining affixes and other mislemmatizations. We collapse numbers into a single number symbol, remove special characters, and cluster the remaining lemmas into 10 000 clusters with word2vec (Mikolov et al., 2013).

5 Training details

We use the following parameters for the network: weight of auxiliary task between 0.001 and 0.75, 64 dimensional character embeddings, 256 dimensional word embeddings, 128 dimensional aux embeddings, 2*256 dimensional encoder state, 1024 dimensional word decoder state, 1024 dimensional character decoder state, 256 dimensional attention, everything except 25k most frequent source words embedded by character level encoder, 50k most frequent target words output by word level decoder, 10k overlap between word level and character level vocabularies during training.

For training, we use Adam with initial learning rate 0.001 and gradient norm clipped to 5.0.

The systems have been tuned towards characterF-1.0 (Popovic, 2015, 2016). We optimize the beam search parameters, using a grid search. The optimal parameters were α 0.012,

β 0.3, γ 0.2, λ 3, pruning margin 1.4, and weight 0.8 for the character-level cost.

We use an ensemble procedure, in which the combined prediction is computed as the mean after the softmax layer of the predictions of 4 models. The primary system uses systems from 4 runs with different weights for the auxiliary task. The systems trained for comparison—a subword system based on Morfessor FlatCat and the systems in ablation experiments—were ensembled using 4 save points from a single run.

To include an example of subword NMT, we also submit our FlatCat system. As preprocessing, the target side has been segmented using Morfessor FlatCat (Grönroos et al., 2014), which was tuned to produce a subword lexicon of approximately 60k symbols. Segmenting names into characters is applied in addition to the FlatCat segmentation. The FlatCat segmented system uses WMT 2016 data only, i.e., omits the Rapid corpus.

The FlatCat subword system uses the standard HNMT decoder. It uses neither the hybrid word-character decoder nor MTL. We did however use the improved beam search with penalties.

6 Results

We evaluate the systems using characterF with β set to 1.0 and 2.0, and cased BLEU using the `mteval-v13a.pl` script. We also include Translation Error Rate (TER) results for the submitted systems. Our primary system has the best TER score of all participants.

As the development test set we use both reference translations of the newstest 2016 set. Table 1 shows the submitted ensemble systems, and the best single model for our primary system. As our system has a tendency to repeat certain words, we also evaluate the primary system after a post-processing step in which consecutive repetitions are removed.

We perform ablation experiments for all new components in our system, by removing each of them separately (non-cumulative effect). Results are shown in Table 2.

All added components were beneficial. The largest improvement, +4.3 BLEU, comes from the attention mechanism in the character decoder, combined with segmenting names into characters.

Multi-task learning improves BLEU by +2.03. Not all auxiliary labels are equally important. PoS tags (+2.17 BLEU) and clustered lemmas (+2.09 BLEU) perform above average, and removing either of them yields worse BLEU than not using MTL at all. The results of both characterF measures differ in this, ranking not using MTL as worse than all the partial MTL variants.

The overattending penalty to the beam search gives a much more modest gain of +0.13 BLEU. The coverage penalty is the most important of the beam search penalties. In total, the beam search heuristics yield an improvement of +1.01 BLEU.

In the human evaluation, our primary system was ranked in the second of five clusters (tied 3rd to 5th place).

7 Discussion

All our added components improved the translation quality.

The largest improvement comes from the modifications intended to enable character-to-character copying: segmenting names into characters and character-level attention. However, the simple heuristic used for selecting words to segment can make translation more difficult in some cases, e.g. the names of institutions are typically capitalized, but translated on a term level. Replacing the heuristic with named-entity recognition or other more advanced methods is left for future work.

A common type of error made by our system is overtranslation through repetition. A possible explanation for the effect is the way that the levels of the hybrid word-character decoder are connected. There is no connection from the character level back to the word level. The surface forms generated by the character-level decoder are conditionally independent given the word-level hidden states, which can be similar to the states at adjacent time steps. The word-level decoder must decide on the number of words in an expression, which is a difficult task if the proportion of <UNK> tokens becomes large. The overattending penalty is only partially successful at reducing the repetition, and increasing the penalty weight deteriorates overall performance before eliminating the problem.

8 Conclusion

Our results show that translation into a morphologically complex language can be improved using word-level labels from morphological analysis combined with a hybrid word-character decoder. Adding an attention mechanism to the character decoder yields a large quality improvement.

Acknowledgements

This research has been supported by the Academy of Finland under the Finnish Centre of Excellence Program 2012–2017 (grant n°251170). Computer resources within the Aalto University School of Science "Science-IT" project were used. We wish to thank Jörg Tiedemann for the useful discussions and for sharing his backtranslations.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR15*.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, Springer, pages 95–133.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. *arXiv preprint arXiv:1702.03525* .

Mark Fishel and Harri Kirik. 2010. Linguistically motivated unsupervised segmentation for machine translation. In *Language Resources and Evaluation (LREC)*. Valletta, Malta.

Mercedes García-Martínez, Loïc Barrault, and Fethi Bougares. 2016. Factored neural machine translation. *arXiv preprint arXiv:1609.04621* .

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. 2015. Tuning phrase-based segmented translation for a morphologically complex target language. In *WMT15*. Association for Computational Linguistics, Lisbon, Portugal, pages 105–111.

Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. 2014. Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *COLING14*. Association for Computational Linguistics, pages 1177–1185.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *EMNLP-CoNLL*. pages 868–876.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *ACL16*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems (NIPS)*. Lake Tahoe, NV, USA, pages 3111–3119.

Maja Popovic. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *WMT15*. pages 392–395.

Maja Popovic. 2016. chrF deconstructed: β parameters and n-gram weights. In *WMT16*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *WMT16*. Association for Computational Linguistics, pages 83–91.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. In *ACL16*.

Miikka Silfverberg, Teemu Ruokolainen, Krister Lindén, and Mikko Kurimo. 2016. FinnPos: an open-source morphological tagging and lemmatization toolkit for Finnish. *Language Resources and Evaluation (LREC)* 50(4):863–878.

Sami Virpioja, Jaakko J Väyrynen, Mathias Creutz, and Markus Sadeniemi. 2007. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. *Machine Translation Summit XI* 2007:491–498.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Publication XI

Stig-Arne Grönroos, Benoit Huet, Mikko Kurimo, Jorma Laaksonen, Bernard Merialdo, Phu Pham, Mats Sjöberg, Umut Sulubacak, Jörg Tiedemann, Raphael Troncy, and Raúl Vázquez. The MeMAD submission to the WMT18 multimodal translation task. In *Proceedings of the Third Conference on Machine Translation*, Brussels, Belgium, pages 603–611, Oct 2018.

The MeMAD Submission to the WMT18 Multimodal Translation Task

Stig-Arne Grönroos
Aalto University

Benoit Huet
EURECOM

Mikko Kurimo
Aalto University

Jorma Laaksonen
Aalto University

Bernard Merialdo
EURECOM

Phu Pham
Aalto University

Mats Sjöberg
Aalto University

Umut Sulubacak
University of Helsinki

Jörg Tiedemann
University of Helsinki

Raphael Troncy
EURECOM

Raúl Vázquez
University of Helsinki

Abstract

This paper describes the MeMAD project entry to the WMT Multimodal Machine Translation Shared Task.

We propose adapting the Transformer neural machine translation (NMT) architecture to a multi-modal setting. In this paper, we also describe the preliminary experiments with text-only translation systems leading us up to this choice.

We have the top scoring system for both English-to-German and English-to-French, according to the automatic metrics for *flickr18*.

Our experiments show that the effect of the visual features in our system is small. Our largest gains come from the quality of the underlying text-only NMT system. We find that appropriate use of additional data is effective.

1 Introduction

In multi-modal translation, the task is to translate from a source sentence and the image that it describes, into a target sentence in another language. As both automatic image captioning systems and crowd captioning efforts tend to mainly yield descriptions in English, multi-modal translation can be useful for generating descriptions of images for languages other than English. In the MeMAD project[1], multi-modal translation is of interest for creating textual versions or descriptions of audio-visual content. Conversion to text enables both indexing for multi-lingual image and video search, and increased access to the audio-visual materials for visually impaired users.

Data set	images	en	de	fr	sentences
Multi30k	✓	✓	✓	✓	29k
MS-COCO	✓	✓	+	+	616k
OpenSubtitles		✓	✓	✓	23M/42M
		1M, 3M, and 6M subsets used.			

Table 1: Summary of data set sizes. ✓means attribute is present in original data. + means data set augmented in this work.

We adapt[2] the Transformer (Vaswani et al., 2017) architecture to use global image features extracted from Detectron, a pre-trained object detection and localization neural network. We use two additional training corpora: MS-COCO (Lin et al., 2014) and OpenSubtitles2018 (Tiedemann, 2009). MS-COCO is multi-modal, but not multi-lingual. We extended it to a synthetic multi-modal and multilingual training set. OpenSubtitles is multilingual, but does not include associated images, and was used as text-only training data. This places our entry in the unconstrained category of the WMT shared task. Details on the architecture used in this work can be found in Section 4.1. Further details on the synthetic data are presented in Section 2. Data sets are summarized in Table 1.

2 Experiment 1: Optimizing Text-Based Machine Translation

Our first aim was to select the text-based MT system to base our multi-modal extensions on.

[1] https://www.memad.eu/

[2] Our fork available from https://github.com/Waino/OpenNMT-py/tree/develop_mmod

Proceedings of the Third Conference on Machine Translation (WMT), Volume 2: Shared Task Papers, pages 603–611

https://doi.org/10.18653/v1/W18-64066

EN-FR	flickr16	flickr17	mscoco17
multi30k	61.4	54.0	43.1
+SUBS$_{\text{full}}$	53.7	48.9	47.0
+domain-tuned	66.1	59.7	**51.7**
+ensemble-of-3	**66.5**	**60.2**	51.6
EN-DE	flickr16	flickr17	mscoco17
multi30k	38.9	32.0	27.7
+SUBS$_{\text{full}}$	41.3	34.1	31.3
+domain-tuned	43.3	38.4	35.0
+ensemble-of-3	**43.9**	**39.6**	**37.0**

Table 2: Adding subtitle data and domain tuning for image caption translation (BLEU% scores). All results with Marian Amun.

	EN-FR	flickr16	flickr17	mscoco17
A	SUBS1M$_H$+MS-COCO	66.3	60.5	52.1
A	+domain-tuned	66.8	60.6	52.0
A	+labels	**67.2**	60.4	51.7
T	SUBS1M$_{LM}$+MS-COCO	66.9	60.3	**52.8**
T	+labels	**67.2**	**60.9**	52.7
	EN-DE	flickr16	flickr17	mscoco17
A	SUBS1M$_H$+MS-COCO	43.1	39.0	35.1
A	+domain-tuned	43.9	39.4	35.8
A	+labels	43.2	39.3	34.3
T	SUBS1M$_{LM}$+MS-COCO	**44.4**	39.4	35.0
T	+labels	44.1	**39.8**	**36.5**

Table 3: Using automatically translated image captions and domain labels (BLEU% scores). A is short for Amun, T for Transformer.

We tried a wide range of models, but only include results with the two strongest systems: Marian NMT with the *amun* model (Junczys-Dowmunt et al., 2018), and OpenNMT (Klein et al., 2017) with the *Transformer* model.

We also studied the effect of additional training data. Our initial experiments showed that movie subtitles and their translations work rather well to augment the given training data. Therefore, we included parallel subtitles from the OpenSubtitles2018 corpus to train better text-only MT models. For these experiments, we apply the Marian amun model, an attentional encoder-decoder model with bidirectional LSTM's on the encoder side. In our first series of experiments, we observed that domain-tuning is very important when using Marian. The domain-tuning was accomplished by a second training step on in-domain data after training the model on the entire data set. Table 2 shows the scores on development data. We also tried decoding with an ensemble of three independent runs, which also pushed the performance a bit.

Furthermore, we tried to artificially increase the amount of in-domain data by translating existing English image captions to German and French. For this purpose, we used the large MS-COCO data set with its 100,000 images that have five image captions each. We used our best multidomain model (see Table 2) to translate all of those captions and used them as additional training data. This procedure also transfers the knowledge learned by the multidomain model into the caption translations, which helps us to improve the coverage of the system with less out-of-domain data. Hence, we filtered the large collection of translated movie subtitles to a smaller portion of reliable sentence pairs (one million in the experiment we report) and could train on a smaller data set with better results.

We experimented with two filtering methods. Initially, we implemented a basic heuristic filter (SUBS$_H$), and later we improved on this with a language model filter (SUBS$_{LM}$). Both procedures consider each sentence pair, assign it a quality score, and then select the highest scoring 1, 3, or 6 million pairs, discarding the rest. The SUBS$_H$ method counts terminal punctuation (*'.', '...', '?', '!'*) in the source and target sentences, initializing the score as the negative of the absolute value of the difference between these counts. Afterwards, it further decrements the score by 1 for each occurrence of terminal punctuation beyond the first in each of the sentences. The SUBS$_{LM}$ method first preprocesses the data by filtering samples by length and ratio of lengths, applying a rule-based noise filter, removing all characters not present in the Multi30k set, and deduplicating samples. Afterwards, target sentences in the remaining pairs are scored using a character-based deep LSTM language model trained on the Multi30k data. Both selection procedures are intended for noise filtering, and SUBS$_{LM}$ additionally acts as domain adaptation. Table 3 lists the scores we obtained on development data.

To make a distinction between automatically translated captions, subtitle translations and human-translated image captions, we also

introduced domain labels that we added as special tokens to the beginning of the input sequence. In this way, the model can use explicit information about the domain when deciding how to translate given input. However, the effect of such labels is not consistent between systems. For Marian amun, the effect is negligible as we can see in Table 3. For the Transformer, domain labels had little effect on BLEU but were clearly beneficial according to chrF-1.0.

2.1 Preprocessing of textual data

The final preprocessing pipeline for the textual data consisted of lowercasing, tokenizing using Moses, fixing double-encoded entities and other encoding problems, and normalizing punctuation. For the OpenSubtitles data we additionally used the SUBS$_{LM}$ subset selection.

Subword decoding has become popular in NMT. Careful choice of translation units is especially important as one of the target languages of our system is German, a morphologically rich language. We trained a shared 50k subword vocabulary using Byte Pair Encoding (BPE) (Sennrich et al., 2015). To produce a balanced multi-lingual segmentation, the following procedure was used: First, word counts were calculated individually for English and each of the 3 target languages Czech[3], French and German. The counts were normalized to equalize the sum of the counts for each language. This avoided imbalance in the amount of data skewing the segmentation in favor of some language. Segmentation boundaries around hyphens were forced, overriding the BPE.

Multi-lingual translation with target-language tag was done following Johnson et al. (2016). A special token, e.g. <TO_DE> to mark German as the target language, was prefixed to each paired English source sentence.

3 Experiment 2: Adding Automatic Image Captions

Our first attempt to add multi-modal information to the translation model includes the incorporation of automatically created image captions in a purely text-based translation engine. For this, we generated five English captions for each of the images in the provided training and test data. This was done by using our in-house captioning system (Shetty et al., 2018). The image captioning system uses a 2-layer LSTM with residual connections to generate captions based on scene context and object location descriptors, in addition to standard CNN-based features. The model was trained with the MS-COCO training data and used to be state of the art in the COCO leaderboard[4] in Spring 2016. The beam search size was set to five.

EN-FR	flickr16	flickr17	mscoco17
multi30k	61.4	54.0	43.1
+autocap (dual attn.)	60.9	52.9	43.3
+autocap 1 (concat)	61.7	53.7	43.9
+autocap 1-5 (concat)	**62.2**	**54.4**	**44.1**
EN-DE	**flickr16**	**flickr17**	**mscoco17**
multi30k	38.9	32.0	27.7
+autocap (dual attn.)	37.8	30.2	27.0
+autocap 1 (concat)	39.7	**32.2**	**28.8**
+autocap 1-5 (concat)	**39.9**	32.0	28.7

Table 4: Adding automatic image captions (only the best one or all 5). The table shows BLEU scores in %. All results with Marian Amun.

We tried two models for the integration of those captions: (1) a dual attention multi-source model that adds another input sequence with its own decoder attention and (2) a concatenation model that adds auto captions at the end of the original input string separated by a special token. In the second model, attention takes care of learning how to use the additional information and previous work has shown that this, indeed, is possible (Niehues et al., 2016; Östling et al., 2017). For both models, we applied Marian NMT that already includes a working implementation of dual attention translations. Table 4 summarizes the scores on the three development test sets for English-French and English-German.

We can see that the dual attention model does not work at all and the scores slightly drop. The concatenation approach works better probably because the common attention

[3]Czech was later dropped as a target language due to time constraints.

[4]https://competitions.codalab.org/competitions/3221

model learns interactions between the different types of input. However, the improvements are small if any and the model basically learns to ignore the auto captions, which are often very different from the original input. The attention pattern in the example of Figure 1 shows one of the very rare cases where we observe at least some attention to the automatic captions.

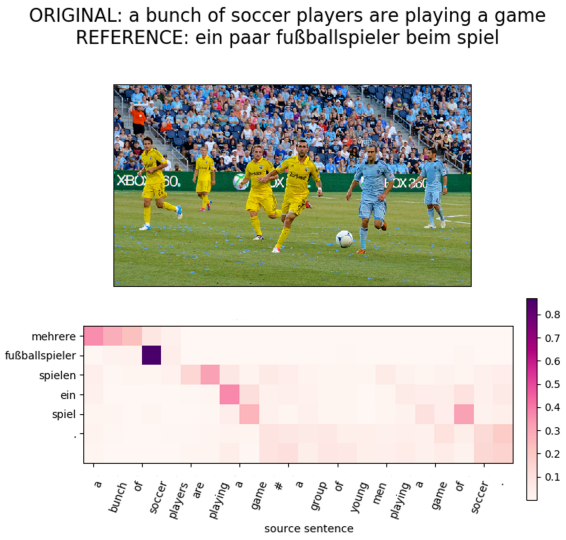


Figure 1: Attention layer visualization for an example where at least one of the attention weights for the last part of the sentence, which corresponds to the automatically generated captions, obtains a value above 0.3

4 Experiment 3: Multi-modal Transformer

One benefit of NMT, in addition to its strong performance, is its flexibility in enabling different information sources to be merged. Different strategies to include image features both on the encoder and decoder side have been explored. We are inspired by the recent success of the Transformer architecture to adapt some of these strategies for use with the Transformer.

Recurrent neural networks start their processing from some **initial hidden state**. Normally, a zero vector or a learned parameter vector is used, but the initial hidden state is also a natural location to introduce additional context e.g. from other modalities. Initializing can be applied in either the encoder (IMG_E) or decoder (IMG_D) (Calixto et al., 2017). These approaches are not directly applicable to the Transformer, as it is not a recurrent model, and lacks a comparable initial hidden state.

Double attention is another popular choice, used by e.g. Caglayan et al. (2017). In this approach, two attention mechanisms are used, one for each modality. The attentions can be separate or hierarchical. While it would be possible to use double attention with the Transformer, we did not explore it in this work. The multiple multi-head attention mechanisms in the Transformer leave open many challenges in how this integration would be done.

Multi-task learning has also been used, e.g. in the Imagination model (Elliott and Kádár, 2017), where the auxiliary task consists of reconstructing the visual features from the source encoding. Imagination could also have been used with the Transformer, but we did not explore it in this work.

The **source sequence** itself is also a possible location for including the visual information. In the IMG_W approach, the visual features are encoded as a pseudo-word embedding concatenated to the word embeddings of the source sentence. When the encoder is a bidirectional recurrent network, as in Calixto et al. (2017), it is beneficial to add the pseudo-word both at the beginning and the end to make it available for both encoder directions. This is unnecessary in the Transformer, as it has equal access to all parts of the source in the deeper layers of the encoder. Therefore, we add the pseudo-word only to the beginning of the sequence. We use an affine projection of the image features $V \in \mathbb{R}^{80}$ into a pseudo-word embedding $x_I \in \mathbb{R}^{512}$

$$x_I = W_{src} \cdot V + b_I.$$

In the LIUM *trg-mul* (Caglayan et al., 2017), the **target embeddings** and visual features are interacted through elementwise multiplication.

$$y'_j = y_j \odot \tanh(W^{dec}_{mul} \cdot V)$$

Our initial gating approach resembles *trg-mul*.

4.1 Architecture

The baseline NMT for this experiment is the OpenNMT implementation of the Transformer. It is an encoder-decoder NMT system

using the Transformer architecture (Vaswani et al., 2017) for both the encoder and decoder side. The Transformer is a deep, non-recurrent network for processing variable-length sequences. A Transformer is a stack of layers, consisting of two types of sub-layer: multi-head (MH) attention (Att) sub-layers and feed-forward (FF) sub-layers:

$$\begin{aligned} \text{Att}(Q,K,V) &= \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \\ a_i &= \text{Att}(QW_i^Q, KW_i^K, VW_i^V) \\ \text{MH}(Q,K,V) &= [a_1;\ldots;a_h]W^O \\ \text{FF}(x) &= \max(0, xW_1 + b_1)W_2 + b_2 \end{aligned} \quad (1)$$

where Q is the input query, K is the key, and V the attended values. Each sub-layer is individually wrapped in a residual connection and layer normalization.

When used in translation, Transformer layers are stacked into an encoder-decoder structure. In the encoder, the layer consists of a self-attention sub-layer followed by a FF sub-layer. In self-attention, the output of the previous layer is used as queries, keys and values $Q = K = V$. In the decoder, a third context attention sub-layer is inserted between the self-attention and the FF. In context attention, Q is again the output of the previous layer, but $K = V$ is the output of the encoder stack. The decoder self-attention is also masked to prevent access to future information. Sinusoidal position encoding makes word order information available.

Decoder gate. Our first approach is inspired by *trg-mul*. A gating layer is introduced to modify the pre-softmax prediction distribution. This allows visual features to directly suppress a part of the output vocabulary. The probability of correctly translating a source word with visually resolvable ambiguity can be increased by suppressing the unwanted choices.

At each timestep the decoder output s_j is projected to an unnormalized distribution over the target vocabulary.

$$y_j = W \cdot s_j + b$$

Before normalizing the distribution using a softmax layer, a gating layer can be added.

$$\begin{aligned} g &= \sigma(W_{gate}^{dec} \cdot V + b_{gate}^{dec}) \\ y'_j &= y_j \odot g \end{aligned} \quad (2)$$

Preliminary experiments showed that gating based on only the visual features did not work. Suppressing the same subword units during the entire decoding of the sentence was too disruptive. We addressed this by using the decoder hidden state as additional input to control the gate. This causes the vocabulary suppression to be time dependent.

$$g_j = \sigma(U_{gate}^{dec} \cdot s_j + W_{gate}^{dec} \cdot V + b_{gate}^{dec}) \quad (3)$$

Encoder gate. The same gating procedure can also be applied to the output of the encoder. When using the encoder gate, the encoded source sentence is disambiguated, instead of suppressing part of the output vocabulary.

$$\begin{aligned} g_i &= \sigma(U_{gate}^{enc} \cdot h_i + W_{gate}^{enc} \cdot V + b_{gate}^{enc}) \\ h'_i &= h_i \odot g_i \end{aligned} \quad (4)$$

The gate biases b_{gate}^{dec} and b_{gate}^{enc} should be initialized to positive values, to start training with the gates opened. We also tried combining both forms of gating.

EN-FR	flickr16	flickr17	mscoco17
IMG_W	*68.30*	**62.45**	52.86
enc-gate	68.01	61.38	**53.40**
dec-gate	67.99	61.53	52.38
enc-gate + dec-gate	**68.58**	*62.14*	*52.98*
EN-DE	**flickr16**	**flickr17**	**mscoco17**
IMG_W	*45.09*	40.81	36.94
enc-gate	44.75	**41.44**	**37.76**
dec-gate	**45.21**	40.79	36.47
enc-gate + dec-gate	44.91	*41.06*	*37.40*

Table 5: Comparison of strategies for integrating visual information (BLEU% scores). All results using Transformer, Multi30k+MS-COCO+SUBS3M$_{LM}$, Detectron mask surface, and domain labeling.

4.2 Visual feature selection

Image feature selection was performed using the LIUM-CVC translation system (Caglayan et al., 2017) training on the WMT18 training

EN-FR	flickr16	flickr17	mscoco17
SUBS3M$_{LM}$ detectron	68.30	62.45	52.86
+ensemble-of-3	68.72	62.70	53.06
−visual features	**68.74**	**62.71**	53.14
−MS-COCO	67.13	61.17	**53.34**
−multi-lingual	68.21	61.99	52.40
SUBS6M$_{LM}$ detectron	68.29	61.73	53.05
SUBS3M$_{LM}$ gn2048	67.74	61.78	52.76
SUBS3M$_{LM}$ text-only	67.72	61.75	53.02
EN-DE	flickr16	flickr17	mscoco17
SUBS3M$_{LM}$ detectron	45.09	40.81	36.94
+ensemble-of-3	45.52	**41.84**	**37.49**
−visual features	**45.59**	41.75	37.43
−MS-COCO	45.11	40.52	36.47
−multi-lingual	44.95	40.09	35.28
SUBS6M$_{LM}$ detectron	45.50	41.01	36.81
SUBS3M$_{LM}$ gn2048	45.38	40.07	36.82
SUBS3M$_{LM}$ text-only	44.87	41.27	36.59
+multi-modal finetune	44.56	41.61	36.93

Table 6: Ablation experiments (BLEU% scores). The row SUBS3M$_{LM}$ *detectron* shows our best single model. Individual components or data choices are varied one by one. + stands for adding a component, and − for removing a component or data set. Multiple modifications are indicated by increasing the indentation.

data, and evaluating on the *flickr16, flickr17* and *mscoco17* data sets. This setup is different from our final NMT architecture as the visual feature selection stage was performed at an earlier phase of our experiments. However, the LIUM-CVC setup without training set expansion was also faster to train which enabled a more extensive feature selection process.

We experimented with a set of state-of-the-art visual features, described below.

CNN-based features are 2048-dimensional feature vectors produced by applying reverse spatial pyramid pooling on features extracted from the 5^{th} Inception module of the pre-trained GoogLeNet (Szegedy et al., 2015). For a more detailed description, see (Shetty et al., 2018). These features are referred to as gn2048 in Table 6.

Scene-type features are 397-dimensional feature vectors representing the association score of an image to each of the scene types in SUN397 (Xiao et al., 2010). Each association score is determined by a separate Radial Basis Function Support Vector Machine (RBF-SVM) classifier trained from pre-trained GoogLeNet CNN features (Shetty et al., 2018).

Action-type features are 40-dimensional feature vectors created with RBF-SVM classifiers similarly to the scene-type features, but using the Stanford 40 Actions dataset (Yao et al., 2011) for training the classifiers. Pre-trained GoogLeNet CNN features (Szegedy et al., 2015) were again used as the first-stage visual descriptors.

Object-type and location features are generated using the Detectron software[5] which implements Mask R-CNN (He et al., 2017) with ResNeXt-152 (Xie et al., 2017) features. Mask R-CNN is an extension of Faster R-CNN object detection and localization (Ren et al., 2015) that also generates a segmentation mask for each of the detected objects. We generated an 80-dimensional *mask surface* feature vector by expressing the image surface area covered by each of the MS-COCO classes based on the detected masks.

We found that the Detectron mask surface resulted in the best BLEU scores in all evaluation data sets for improving the German translations. Only for *mscoco17* the results could be slightly improved with a fusion of mask surface and the SUN 397 scene-type feature. For French, the results were more varied, but we focused on improving the German translation results as those were poorer overall. We experimented with different ways of introducing the image features into the translation model implemented in LIUM-CVC, and found as in (Caglayan et al., 2017), that *trg-mul* worked best overall.

Later we learned that the *mscoco17* test set has some overlap with the COCO 2017 training set, which was used to train the Detectron models. Thus, the results on that test set may not be entirely reliable. However, we still feel confident in our conclusions as they are also confirmed by the *flickr16* and *flickr17* test sets.

4.3 Training

We use the following parameters for the network:[6] 6 Transformer layers in both encoder and decoder, 512-dimensional word embeddings and hidden states, dropout 0.1, batch

[5] https://github.com/facebookresearch/Detectron

[6] Parameters were chosen following the OpenNMT FAQ http://opennmt.net/OpenNMT-py/FAQ.html#how-do-i-use-the-transformer-model

Figure 2: Image 117 was translated correctly as feminine "eine besitzerin steht still und ihr brauner hund rennt auf sie zu ." when not using the image features, but as masculine "ein besitzer …" when using them. The English text contains the word "her". The person in the image has short hair and is wearing pants.

size 4096 tokens, label smoothing 0.1, Adam with initial learning rate 2 and β_2 0.998.

For decoding, we use an ensemble procedure, in which the predictions of 3 independently trained models are combined by averaging after the softmax layer to compute combined prediction.

We evaluate the systems using uncased BLEU using multibleu. During tuning, we also used characterF (Popovic, 2015) with β set to 1.0.

There are no images paired with the sentences in OpenSubtitles. When using OpenSubtitles in training multi-modal models, we feed in the mean vector of all visual features in the training data as a dummy visual feature.

4.4 Results

Based on the previous experiments, we chose the Transformer architecture, Multi30k+MS-COCO+SUBS3M$_{LM}$ data sets, Detectron mask surface visual features, and domain labeling.

Table 5 shows the BLEU scores for this configuration with different ways of integrating the visual features. The results are inconclusive. The ranking according to chrF-1.0 was not any clearer. Considering the results as a whole and the simplicity of the method, we chose IMG$_W$ going forward.

Table 6 shows results of ablation experiments removing or modifying one component or data choice at a time, and results when using ensemble decoding. Using ensemble decoding gave a consistent but small improvement. Multi-lingual models were clearly better than mono-lingual models. For French, 6M sentences of subtitle data gave worse results than 3M.

We experimented with adding multi-modality to a pre-trained text-only system using a fine tuning approach. In the fine tuning phase, a *dec-gate* gating layer was added to the network. The parameters of the main network were frozen, allowing only the added gating layer to be trained. Despite the freezing, the network was still able to unlearn most of the benefits of the additional text-only data. It appears that the output vocabulary was reduced back towards the vocabulary seen in the multi-modal training set. When the experiment was repeated so that the fine-tuning phase included the text-only data, the performance returned to approximately the same level as without tuning (+multi-modal finetune row in Table 6).

To explore the effect of the visual features on the translation of our final model, we performed an experiment where we retranslated using the ensemble while "blinding" the model. Instead of feeding in the actual visual features for the sentence, we used the mean vector of all visual features in the training data. The results are marked *-visual features* in Table 6. The resulting differences in the translated sentences were small, and mostly consisted of minor variations in word order. BLEU scores for French were surprisingly slightly improved by this procedure. We did not find clear examples of successful disambiguation. Figure 2 shows one example of a detrimental use of visual features.

It is possible that adding to the training data forward translations of MS-COCO captions from a text-only translation system introduced a biasing effect. If there is translational ambiguity that should be resolved using the image, the text-only system will not be able to resolve it correctly, instead likely yielding the word that is most frequent in that textual context. Using such data for training a multi-modal system might bias it towards ignoring the image.

On this year's *flickr18* test set, our system scores 38.54 BLEU for English-to-German and 44.11 BLEU for English-to-French.

5 Conclusions

Although we saw an improvement from incorporating multi-modal information, the improvement is modest. The largest differences in quality between the systems we experimented with can be attributed to the quality of the underlying text-only NMT system.

We found the amount of in-domain training data and multi-modal training data to be of great importance. The synthetic MS-COCO data was still beneficial, despite being forward translated, and the visual features being overconfident due to being extracted from a part of the image classifier training data.

Even after expansion with synthetic data, the available multi-modal data is dwarfed by the amount of text-only data. We found that movie subtitles worked well for this purpose. When adding text-only data, domain adaptation was important, and increasing the size of the selection met with diminishing returns.

Current methods do not fully address the problem of how to efficiently learn from both large text-only data and small multi-modal data simultaneously. We experimented with a fine tuning approach to this problem, without success.

Although the effect of the multi-modal information was modest, our system still had the highest performance of the task participants for the English-to-German and English-to-French language pairs, with absolute differences of +6.0 and +3.5 BLEU%, respectively.

Acknowledgments

This work has been supported by the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 780069, and by the Academy of Finland in the project 313988. In addition the Finnish IT Center for Science (CSC) provided computational resources. We would also like to acknowledge the support by NVIDIA and their GPU grant.

References

Ozan Caglayan, Walid Aransa, Adrien Bardet, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, Marc Masana, Luis Herranz, and Joost Van de Weijer. 2017. LIUM-CVC submissions for WMT17 multimodal translation task. In *Proceedings of the Second Conference on Machine Translation*.

Iacer Calixto, Koel Dutta Chowdhury, and Qun Liu. 2017. DCU system report on the WMT 2017 multi-modal machine translation task. In *Proceedings of the Second Conference on Machine Translation*. pages 440–444.

Desmond Elliott and Àkos Kádár. 2017. Imagination improves multimodal translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 130–141.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, pages 2980–2988.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558* .

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*. Association for Computational Linguistics, Melbourne, Australia, pages 116–121. http://www.aclweb.org/anthology/P18-4020.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*. https://doi.org/10.18653/v1/P17-4012.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. *CoRR* abs/1405.0312. http://arxiv.org/abs/1405.0312.

Jan Niehues, Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2016. Pre-translation for neural machine translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 1828–1836.

Robert Östling, Yves Scherrer, Jörg Tiedemann, Gongbo Tang, and Tommi Nieminen. 2017. The helsinki neural machine translation system. In *Proceedings of the Second Conference on Machine Translation*. pages 338–347.

Maja Popovic. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *WMT15*. pages 392–395.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems (NIPS)*. pages 91–99.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. In *ACL16*.

Rakshith Shetty, Hamed Rezazadegan Tavakoli, and Jorma Laaksonen. 2018. Image and video captioning with augmented neural architectures. *IEEE MultiMedia* To appear.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 1–9.

Jörg Tiedemann. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Advances in Natural Language Processing*, John Benjamins, Amsterdam/Philadelphia, Borovets, Bulgaria, volume V, pages 237–248.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. pages 6000–6010.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. 2010. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE conference on Computer vision and pattern recognition (CVPR)*. IEEE, pages 3485–3492.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pages 5987–5995.

Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas J. Guibas, and Fei-Fei Li. 2011. Human action recognition by learning bases of action attributes and parts. In *International Conference on Computer Vision (ICCV)*. Barcelona, Spain, pages 1331–1338.

Publication XII

Sami Virpioja and Stig-Arne Grönroos. LeBLEU: N-gram-based translation evaluation score for morphologically complex languages. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal, pages 411–416, Sep 2015.

LeBLEU: N-gram-based Translation Evaluation Score for Morphologically Complex Languages

Sami Virpioja
Lingsoft, Inc.
Helsinki, Finland
sami.virpioja@lingsoft.fi

Stig-Arne Grönroos
Aalto University
Dept. of Signal Processing and Acoustics
stig-arne.gronroos@aalto.fi

Abstract

This paper describes the LeBLEU evaluation score for machine translation, submitted to WMT15 Metrics Shared Task. LeBLEU extends the popular BLEU score to consider fuzzy matches between word n-grams. While there are several variants of BLEU that allow to non-exact matches between words either by character-based distance measures or morphological preprocessing, none of them use fuzzy comparison between longer chunks of text. The results on WMT data sets show that fuzzy n-gram matching improves correlations to human evaluation especially for highly compounding languages.

1 Introduction

The quality of machine translation has improved to the level that the translation hypotheses are useful starting points for human translators for almost any language pair. In the post-editing task, the ultimate way to evaluate the machine translation quality is to measure the editing time. Editing times are naturally related to the number and types of the edits—and thus the number of keystrokes (Frederking and Nirenburg, 1994)—the post-editor needs to get the final translation from the hypothesis. If we compare the raw translation hypothesis and its post-edited version, an appropriate edit distance measure should correlate to the edit time. However, implementing such a measure is far from trivial.

In automatic speech recognition, common evaluation measures are Word Error Rate (WER) and Letter Error Rate (LER) that are based on the Levenshtein edit distance (Levenshtein, 1966). LER is more reasonable measure than WER for morphologically complex languages, in which the same word can occur in many inflected and derived forms (Creutz et al., 2007). However, both give too high penalty for the variations in word ordering, which are frequent in translations. Even in English, there are often at least two grammatically correct orders for a complex sentence. For languages in which the grammatical roles are marked by morphology and not the word order, there may be many more options.

An edit distance measure suitable for machine translation would require move operations. However, such measures are computationally very expensive: finding the minimum edit distance with moves is NP-hard (Shapira and Storer, 2002), making it cumbersome for evaluation and unsuitable for automatic tuning of the translation models. Possible solutions include limiting the move operations or searching only for an approximate solution. For example, Translation Edit Rate (TER) by Snover et al. (2006) uses a shift operation that moves a contiguous sequence of words to another location, as well as a greedy search algorithm to find the minimum distance. Stanford Probabilistic Edit Distance Evaluation (SPEDE) by Wang and Manning (2012) applies a probabilistic push-down automaton that captures non-nested, limited distance word swapping.

A different approach to avoid the requirement of exactly same word order in the hypothesis and reference translations is to concentrate on comparing only small parts of the full texts. For example, the popular BLEU metric by Papineni et al. (2002) considers only local ordering of words. To be precise, it calculates the geometric mean precision of the n-grams of length between one and four. As high precision is easy to obtain by providing a very short hypothesis translation, hypotheses that are shorter than the reference are penalized by a brevity penalty.

BLEU, TER and many other word-based methods assume that a single word (or n-gram) is either correct or incorrect, nothing in between. This

Proceedings of the Tenth Workshop on Statistical Machine Translation, pages 411–416,
Lisboa, Portugal, 17-18 September 2015.

is problematic for inflected or derived words (e.g. "translate" and "translated" are considered two different words) as well as compound words (e.g. "salt-and-pepper" vs. "salt and pepper"). This is a minor issue for English, but it makes the evaluation unreliable for many other languages. For example, in English–German translation, producing "Arbeits Geberverband" from "employers' organization" would give no hits if the reference had the compound "Arbeitgeberverband".

A common approach to the problem of inflected word forms—as well as to the simpler issues of uppercase letters and punctuation characters—is preprocessing. For example, METEOR (Banerjee and Lavie, 2005; Denkowski and Lavie, 2011) uses a stemmer. Popović (2011) applies and combines BLEU-style scores based on part-of-speech (POS) tags as well as morphemes induced by the unsupervised method by Creutz and Lagus (2005). Also the AMBER score by Chen and Kuhn (2011) combines many BLEU variants, and in some variants, the words are heuristically segmented.

Our approach is to extend the BLEU metric to work better on morphologically complex languages without using any language-specific resources. Instead of giving one point for exactly same n-gram or zero points for any difference, we include "soft" or "fuzzy" hits for word n-grams based on letter edit distance. We call the score LeBLEU; this name can be interpreted either as "Letter-edit-BLEU" or "Levenshtein-BLEU". LeBLEU has two main parameters, n-gram length and fuzzy match threshold, that are easy to tune for different types of languages.[1]

There are at least three previous approaches that resemble LeBLEU in that they try not to overpenalize different word orderings and word forms, but do not require any preprocessing tools or resources. Denoual and Lepage (2005) simply use the standard BLEU score on the level of characters, treating word delimiters as any other characters. In order to capture long enough sequences of text, they increase the maximum n-gram length to 18. Compared to word-based BLEU, their method does not increase the correlations to human evaluation in English.

Homola et al. (2009) propose a score that is a weighted combination of two measures: an alignment score that applies letter edit distances between the word forms and a structural score that measures the differences in word order. In contrast to LeBLEU, it still strongly penalizes errors in compounding, as the alignment is word-to-word and fuzzy matches are accepted only if the LER between a pair of words is lower than 15%.

More recently, Libovický and Pecina (2014) have proposed "tolerant BLEU", a variant of BLEU that similarly to LeBLEU finds fuzzy matches between hypothesis and reference words. Instead of Levenshtein edit distance, they apply a specific affix distance measure that requires an exact match in the middle of the words. Moreover, they apply a more complex procedure, in which the words between the hypothesis and reference are first aligned using the Munkres algorithm. Then the hypothesis words are replaced by the matched reference words while applying a penalty based on the affix distance, and finally standard BLEU calculations are performed on the modified hypothesis. Similarly to the method by Homola et al. (2009), there is no matching between word n-grams of different lengths.

2 Method

LeBLEU differs from the standard BLEU (Papineni et al., 2002) in the following aspects (in the order of decreasing importance):

First, the matching of word n-grams is fuzzy: for a close match, the hits are increased according to a similarity score. The similarity score is one minus letter edit distance normalized by the length of the longer n-gram in characters. Even though we use the term "letter edit", the calculations are based on all characters, including the spaces between the words. If the similarity score is lower than the selected threshold parameter δ, the fuzzy match is ignored.

In contrast to standard BLEU, there is no need for lowercasing or even tokenization. For example, a punctuation character following a word is included in the n-gram as a part of the word. Thus, with a reasonably low threshold parameter, missing the punctuation character will result only in a relative small decrease in the score.

Second, to facilitate the matching of compound words, the hypothesis n-grams are not matched only to reference n-grams of the same order, but n-grams of any order between one and $2n$, where n is highest order of hypothesis n-gram considered.

Third, the brevity penalty is not based on the

[1] In contrast, for example the AMBER score by Chen and Kuhn (2011) includes nearly 20 weight parameters.

number of word tokens but the number of characters in the data. By this, we try to avoid giving too much penalty for mistakes in compound words. Character-based penalty is also one of the penalty variants in AMBER (Chen and Kuhn, 2011).

Fourth, when calculating mean over the different n-gram orders, arithmetic mean is taken instead of geometric mean. That the arithmetic mean is often a better choice than the geometric mean has been noted also by Song et al. (2013).

2.1 Algorithm

Our algorithm for calculating the LeBLEU score consists of four phases: First, the hypothesis n-grams and their frequencies are collected. Second, hypothesis n-grams are matched to the reference n-grams, collecting the normalized letter-edit scores. Third, the scores are summed up for each n-gram order and normalized by the total number of hypothesis n-grams. Finally, average precision over n-gram orders is calculated and multiplied by the brevity penalty.

Only the second phase differs significantly from calculating the standard BLEU score. It is also the most time-consuming part of the algorithm, so we will describe the implemented optimizations in more detail. We also discuss how further speed-up can be obtained by sampling the hypothesis n-grams in the first phase.

2.1.1 Calculating distances between n-grams

As we need to compare all hypothesis n-grams (up to n) to all reference n-grams (up to $2n$), the worst-case complexity for the number of Levenshtein calculations is $O(n^2HR)$ for hypothesis sentence of H words, reference sentence of R words and maximum n-gram order n. We use several strategies to optimize this task without changing the resulting scores.

To calculate the Levenshtein distances, we use a modified version of python-Levenshtein, a Python extension module written in C.[2] The number of function calls from Python to C is minimized by passing in two lists of strings to compare: all extracted n-grams from the hypothesis and reference. This strategy results in a large number of comparisons, making it attractive to prune comparisons that will not affect the final score due to the threshold parameter δ.

Two lower bounds for Levenshtein distance were used for pruning. The first lower bound is given by the difference in lengths of the two strings: the number of letter edits is at least the absolute difference of the lengths. The second lower bound is the bag distance (Bartolini et al., 2002), which uses the difference between character histograms calculated from the compared strings. In addition to the lower bounds, we use early stopping of the dynamic programming algorithm for Levenshtein distance, if all possible paths have grown past the pruning threshold.

For each hypothesis n-gram, the pruning threshold is initially set to δ. As we are looking only for the m-best matches (where m is the number of times the hypothesis n-gram occurred in the sentence), we can constraint the threshold whenever better matches to the reference n-grams are found. For example, if the two best matches are required, a third score that is worse than the current second-best cannot affect the score. Keeping track of the desired number of best matches can be accomplished using for example a heap data structure. However, most of the n-grams occur only once, in which case the heap degenerates into a single item. To simplify the implementation, we adjust the threshold only in this case.

2.1.2 Sampling of n-grams

Regardless of the optimizations above, the evaluation speed may get impractically slow for very long sentences. In such cases, a suitable approximation is to estimate the precision for only a subset of the hypothesis n-grams. If the sample size is limited to L n-grams, the time complexity becomes $O(LnR)$. A sensible scheme is to select n-grams evenly from the hypothesis sentence. In practice, we exclude or include n-grams starting from every kth word for a suitable value of k.[3] If the gaps are never longer than $n - 1$ words, all words in the hypothesis will influence the result. We set the maximum n-gram sample size L to 2000. If $n = 4$, this means that we use all n-grams if the number of words in the hypothesis $H \leq 500$. Some words in the hypothesis would be completely discarded only if $H > 2000$.

[2] Our fork is available from `https://github.com/Waino/python-Levenshtein`.

[3] If $L/H < 0.5$, we set $k = \lfloor H/L \rfloor$ and include every kth word. Otherwise we set $k = \lfloor H/(H - L) \rfloor$ and exclude every kth word.

3 Experiments

We study the proposed evaluation score using the data sets from the shared tasks of the Workshops on Statistical Machine Translation (WMT). The data sets contain human evaluations for different machine translation systems and system combination outputs. The translation hypotheses are ranked both in the level of segments (individual sentences) and systems. The translation hypotheses and references were used as inputs to the LeBLEU score as such: no preprocessing was performed on the texts.

3.1 Parameter tuning

We tuned the two parameters of the evaluation score on the data sets published from the WMT 2013 and 2014 shared tasks (Macháček and Bojar, 2013; Macháček and Bojar, 2014). We ran a grid search on the parameters for each language and level. We tested four values of the maximum n-gram length n (from 1 to 4) and six values of the fuzzy match threshold δ (from 0.2 to 0.8 using step size 0.1).

Our WMT 2015 submission includes two versions regarding the method parameters: "default" and "optimized". For the default submission, we selected the parameters based on the smallest rank sum over all languages, data sets (2013/2014) and levels of evaluation (system/segment). These parameters, which we set as the default parameters for our implementation, are $n = 4$ and $\delta = 0.4$.

For the optimized submission, we took the parameters with the best average correlation over WMT 2013 and 2014 data sets for each language pair and level of evaluation. The results are shown in Table 1. For the Finnish language that was not present in the 2013 and 2014 shared tasks, we took the best parameters for German, another language with complex morphology and long compound words.

		segment		system	
Source	Target	n	δ	n	δ
English	French	4	0.7	4	0.4
English	German	3	0.2	4	0.2
English	Czech	2	0.3	4	0.3
English	Russian	2	0.3	2	0.2
French	English	3	0.6	4	0.6
German	English	4	0.5	4	0.4
Czech	English	4	0.5	4	0.7
Russian	English	4	0.5	4	0.3

Table 1: Results of parameter optimization for each language pair and level of evaluation (segment or system).

3.2 Results for the WMT shared tasks

Table 2 shows the results from the WMT 2013, WMT 2014, and WMT 2015. Topline for system-level data of WMT 2013 is not included due to the use of Spearman's rank correlation instead of Pearson's product-moment correlation. Segment-level results of WMT 2013 are dominated by single submission, SIMPBLEU-RECALL by Song et al. (2013). Considering morphologically complex languages, LeBLEU would have ranked first in English–German and second in English–Czech and English–Russian. For translations to English, LeBLEU would have ranked in the top five among the 10 methods.

For WMT 2014 segment-level data, optimized LeBLEU provides the highest correlations for all language pairs from English. It also outperforms all the included methods for English–German and English–Russian system-level data. For system-level English–French, it would have ranked 5th. For system-level English–Czech, the optimized parameters yielded lower correlation than the default ones, and neither come close to the topline. Somewhat surprisingly, LeBLEU provides the top correlation for system-level German–English and third best for system-level Czech–English translations. For other system-level pairs to English, and all segment-level pairs to English, the correlations are reasonably high but quite far from the respective toplines. We can also compare LeBLEU to two related methods, standard BLEU and AMBER (Chen and Kuhn, 2011). LeBLEU outperforms both in almost all tasks already with the default parameters. The only exception is the system-level English–Czech task, in which BLEU provided a slightly higher correlation.

In the WMT 2015 evaluation, LeBLEU provides quite stable correlations across the different language pairs: Segment-level correlations are between 0.345–0.436 with default parameters and 0.347–0.438 with optimized parameters. System-level correlations are between 0.850–0.955 with default parameters and 0.842–0.984 with optimized parameters, except for English–Finnish, which gets 0.835 with the default parameters and

			WMT 2013			WMT 2014					WMT 2015		
Source	Target	Level	def.	opt.	top	def.	opt.	ref-B	ref-A	top	def.	opt.	top
English	French	segment	.231	.234	**.261**	.292	**.296**	.256	.264	.293	.345	.347	**.366**
English	Finnish	segment	–	–	–	–	–	–	–	–	.368	.368	**.380**
English	German	segment	.247	**.260**	.254	**.273**	**.273**	.191	.227	.268	.398	**.399**	.398
English	Czech	segment	.167	.168	**.192**	.342	**.349**	.290	.302	.344	.406	.410	**.446**
English	Russian	segment	.230	.233	**.245**	.446	**.449**	.381	.397	.440	.404	.404	**.439**
French	English	segment	.255	.259	**.303**	.380	.395	.378	.367	**.433**	.373	.376	**.398**
Finnish	English	segment	–	–	–	–	–	–	–	–	.383	.391	**.445**
German	English	segment	.256	.262	**.318**	.324	.320	.271	.313	**.380**	.402	.399	**.482**
Czech	English	segment	.225	.227	**.388**	.278	.282	.213	.246	**.328**	.436	.438	**.495**
Russian	English	segment	.229	.230	**.234**	.302	.309	.263	.294	**.355**	.376	.374	**.418**
English	French	system	.971	.971	–	.947	.947	.937	.928	**.960**	.933	.933	**.964**
English	Finnish	system	–	–	–	–	–	–	–	–	.835	.803	**.878**
English	German	system	.947	.919	–	.451	**.531**	.216	.241	.357	.850	.868	**.879**
English	Czech	system	.842	.857	–	.973	.964	.976	.972	**.988**	.953	.952	**.977**
English	Russian	system	.787	.870	–	.926	**.941**	.915	.926	**.941**	.896	.908	**.970**
French	English	system	.948	.956	–	.964	.964	.952	.948	**.981**	.955	.984	**.997**
Finnish	English	system	–	–	–	–	–	–	–	–	.900	.900	**.977**
German	English	system	.933	.933	–	**.963**	**.963**	.832	.910	.943	.916	.916	**.981**
Czech	English	system	.960	.946	–	.918	.988	.909	.744	**.993**	.947	.976	**.993**
Russian	English	system	.836	.855	–	.805	.799	.789	.797	**.870**	.908	.842	**.981**

Table 2: Performance of LeBLEU in recent WMT metrics shared tasks. Pearson's correlation coefficients (system-level data) and average Kendall's tau correlation coefficients (segment-level data) for LeBLEU with default parameters (def.), LeBLEU with optimized parameters (opt.), and topline method for the shared task (top). For WMT 2014 data, also two reference methods are included: BLEU (ref-B) and AMBER (ref-A).

only 0.803 with the German-optimized parameters. The choice of German-based parameters was clearly unsuccessful, and the effect of optimization for evaluation in Finnish remains to be seen. On average, optimization based on WMT 2013 and 2014 data sets improved the performance.

Compared to other methods submitted to WMT 2015, LeBLEU outperformed others in segment-level English–German translation. It also ranked second in system-level English–German and third in segment-level English–French. Moreover, even though unoptimized for the task, it ranked third in segment-level and fourth in system-level English–Finnish evaluations.

4 Conclusions

We have described the LeBLEU evaluation score for machine translation. It is an extension of the popular BLEU evaluation metric, but much more suitable for evaluating machine translation to morphologically complex languages. The extension is conceptually simple and does not require any language-specific resources. Instead, morphological variants and mistakes in compound words are accepted by using fuzzy matching between the word n-grams in the hypothesis and reference translations.

In the WMT15 shared task, LeBLEU provided high correlations to the human evaluations especially when translating from English to a morphologically more complex language. In particular, it outperformed other methods in the segment-level evaluation of English–German translation. The performance is equally good for WMT 2013 and 2014 data sets. This is remarkable especially as the method uses neither rule-based nor data-driven tools for morphological processing. As German is a highly compounding language, this indicates that the mistakes in compound words are frequently over-penalized by the current evaluation methods.

Implementation for the LeBLEU evaluation score is available from `https://github.com/Waino/LeBLEU`.

Acknowledgments

We thank Vesa Siivola for his help on the initial implementation of the method and useful comments on the manuscript.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. 2002. String matching with metric trees using an approximate distance. In *String processing and information retrieval*, pages 271–283. Springer.

Boxing Chen and Roland Kuhn. 2011. AMBER: A modified BLEU, enhanced ranking metric. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 71–77, Edinburgh, Scotland, July. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. Technical Report A81, Publications in Computer and Information Science, Helsinki University of Technology.

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. 2007. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Transactions on Speech and Language Processing*, 5(1):3:1–3:29, December.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91, Edinburgh, Scotland, July. Association for Computational Linguistics.

Etienne Denoual and Yves Lepage. 2005. BLEU in characters: towards automatic MT evaluation in languages without word delimiters. In *Companion Volume to the Proceedings of the Second International Joint Conference on Natural Language Processing*, pages 81–86.

Robert Frederking and Sergei Nirenburg. 1994. Three heads are better than one. In *Proceedings of the Fourth Conference on Applied Natural Language Processing*, pages 95–100, Stuttgart, Germany, October. Association for Computational Linguistics.

Petr Homola, Vladislav Kuboň, and Pavel Pecina. 2009. A simple automatic MT evaluation metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 33–36, Athens, Greece, March. Association for Computational Linguistics.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Jindřich Libovický and Pavel Pecina. 2014. Tolerant BLEU: a submission to the WMT14 metrics task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 409–413, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

Matouš Macháček and Ondřej Bojar. 2013. Results of the WMT13 metrics shared task. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 45–51, Sofia, Bulgaria, August. Association for Computational Linguistics.

Matouš Macháček and Ondřej Bojar. 2014. Results of the WMT14 metrics shared task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 293–301, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, USA, July. Association for Computational Linguistics.

Maja Popović. 2011. Morphemes and POS tags for n-gram based evaluation metrics. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 104–107, Edinburgh, Scotland, July. Association for Computational Linguistics.

Dana Shapira and James A. Storer. 2002. Edit distance with move operations. In *Proceedings of the 13th Annual Symposium on Computational Pattern Matching*, pages 85–98.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Xingyi Song, Trevor Cohn, and Lucia Specia. 2013. BLEU deconstructed: Designing a better MT evaluation metric, March. On-line: `http://staffwww.dcs.shef.ac.uk/people/X.Song/song13deconstructed.pdf`. Accessed Oct 2013.

Mengqiu Wang and Christopher Manning. 2012. SPEDE: Probabilistic edit distance metrics for MT evaluation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 76–83, Montréal, Canada, June. Association for Computational Linguistics.

In a multilingual world that is rapidly becoming more digital, machine translation is vital both for communication and access to information. As most of the on-line information is in English, the greatest benefits fall to non-English speakers. Languages with few speakers or a low degree of digitalization lack the large amounts of parallel training data needed for current methods. Languages with rich morphology have very large numbers of word forms, with many poorly modeled rare words. Techniques for reducing data requirements make machine translation feasible for these low-resource languages.

This thesis presents improvements to how text is represented as subwords, how languages can benefit from each other through cross-lingual transfer, and how additional monolingual data can be used through denoising auxiliary tasks.

Aalto-DD 202/2020

ISBN 978-952-64-0168-3 (printed)
ISBN 978-952-64-0169-0 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Electrical Engineering
Department of Signal Processing and Acoustics
www.aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS